Bundesamt
für Sicherheit in der
Informationstechnik

Sicherheitsstudie Content Management Systeme (CMS)

Version 1.0

Autoren

Christian Breitenstrom,
Clemens Micklisch,
Dr. Małgorzata Mochól

von der]init[Aktiengesellschaft für digitale Kommunikation
Köpenicker Straße 9
10997 Berlin

Jürgen Baum,
Clemens Pflüger,
Jan Steffan

vom Fraunhofer-Institut für Sichere Informationstechnologie (SIT)
Standort: Schloss Birlinghoven
53754 Sankt Augustin

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de
© Bundesamt für Sicherheit in der Informationstechnik 2013

Inhaltsverzeichnis

1 Einleitung ... 7
1.1 Motivation ... 7
1.2 Untersuchungsgegenstand ... 8
1.3 Untersuchungsmethodik ... 9
1.4 Aufbau der Studie ... 10
1.5 Notation ... 10
2 Einführung in Content Management Systeme (CMS) ... 12
2.1 Technischer und funktionaler Aufbau ... 12
2.2 Beschreibung der ausgewählten Systeme ... 14
2.2.1 Drupal ... 14
2.2.2 Plone ... 17
2.2.3 WordPress ... 19
2.2.4 Joomla! ... 22
2.2.5 TYPO3 ... 25
2.3 Aufstellung der Bewertungskriterien ... 27
2.3.1 Überblick ... 27
2.3.2 Service Design ... 30
2.3.3 Service Transition ... 44
2.3.4 Service Operation ... 47
2.4 Typische Anwendungsszenarien ... 51
2.4.1 Szenario 1: „Private Event Site“ ... 51
2.4.2 Szenario 2: „Bürgerbüro einer kleinen Gemeinde“ ... 51
2.4.3 Szenario 3: „Open Government Site einer Kleinstadt“ ... 52
2.4.4 Szenario 4: „Mittelständisches Unternehmen mit mehreren Standorten“ ... 52
3 Aktuelle Bedrohungslage ... 53
3.1 Problemstellung ... 53
3.2 Erhebung ... 54
3.2.1 Drupal ... 54
3.2.2 Plone ... 56
3.2.3 WordPress ... 58
3.2.4 Joomla! ... 60
3.2.5 TYPO3 ... 62
3.2.6 Basis-Technologien ... 64
3.2.7 Datenbank Systeme ... 65
3.3 Auswertung ... 65
3.3.1 Klassifizierung nach Schwachstellentypen ... 65
3.3.2 Verteilung der Schwachstellen zwischen Basis-Installation und Erweiterung ... 67
3.3.3 Schwachstellen-Kompensation ... 67
3.3.4 Vergleich der CMS zueinander ... 68
3.3.5 Diskussion ... 68
4 Sicherheitsuntersuchung ... 70
4.1 Vorgehen ... 70
4.2 Service Design ... 70

4.2.1 Geschäftsführung (Governance) 70
4.2.2 Prüfung / Test (Verifikation) 72
4.2.3 Softwareentwicklung (Construction) 72
4.3 Service Transition 79
4.3.1 Installation (Deployment) 79
4.3.2 Anwenderdokumentation 80
4.4 Service Operation 82
4.4.1 Marktdurchdringung 82
4.4.2 Behandlung von Änderungen/Security Patches 82
4.4.3 Betrieb 83
4.5 Zusammenfassung der Sicherheitsuntersuchung 84
4.6 Abgleich mit der aktuellen Bedrohungslage 85
5 Zusammenfassung der Studie 87
5.1 Ergebnisse 87
5.1.1 Sicherheitsniveau der betrachteten CMS 87
5.1.2 Optimierungsmöglichkeiten 87
5.2 Szenario-basierte Handlungsempfehlungen 88
5.2.1 Szenario 1: „Private Event Site“ 89
5.2.2 Szenario 2: „Bürgerbüro in einer kleinen Gemeinde“ 89
5.2.3 Szenario 3: „Open Government Site einer Kleinstadt“ 91
5.2.4 Szenario 4: „Mittelständisches Unternehmen mit mehreren Standorten“ 93
5.3 Ausblick 94
5.3.1 Entwicklung der Gefährdungen (OWASP Top 10 2013) 94
5.3.2 Standards für die sichere Konfiguration 95
5.3.3 Alternative Sicherheitsstrategien 95
Anhang: Faktenübersicht 97
Anhang: Empfehlungsübersicht 98
Anhang: Ergebnistabelle TYPO3 CMS 6.0 99
Anhang: Ergebnistabelle WordPress 3.4.2 112
Anhang: Ergebnistabelle Joomla! 3.02 123
Anhang: Ergebnistabelle Drupal 7.17 134
Anhang Ergebnistabelle: Plone 4.21 145
Linkliste der wichtigsten Quellen 162
Stichwortverzeichnis 163
Abkürzungsverzeichnis 165

Abbildungsverzeichnis

Abbildung 2.1: Generisches UML Deployment Diagramm 13
Abbildung 2.2: Drupal – Systemarchitektur 15
Abbildung 2.3: Plone – Systemarchitektur 18
Abbildung 2.4: WordPress – Systemarchitektur 21
Abbildung 2.5: Joomla! – Systemarchitektur 24
Abbildung 2.6: TYPO3 – Systemarchitektur 26
Abbildung 2.7: Kriterienüberblick 28
Abbildung 2.8: Kriterienüberblick – Service Design (Geschäftsführung) 30

Abbildung 2.9: Kriterienüberblick – Service Design (Prüfung, Test) 32
Abbildung 2.10: Kriterienüberblick – Service Design (Softwareentwicklung) 33
Abbildung 2.11: Kriterienüberblick – Service Transition 44
Abbildung 2.12: Kriterienüberblick – Service Operation 47
Abbildung 3.1: Drupal – Schwachstellen: Basis-Installation und Erweiterungen 55
Abbildung 3.2: Plone – Schwachstellen: Basis-Installation und Erweiterungen 57
Abbildung 3.3: WordPress – Schwachstellen: Basis-Installation und Erweiterungen 59
Abbildung 3.4: Joomla! – Schwachstellen: Basis-Installation und Erweiterungen 61
Abbildung 3.5: TYPO3 – Schwachstellen: Basis-Installation und Erweiterungen 63
Abbildung 3.6: Verteilung der gemeldeten Schwachstellen 66
Abbildung 3.7: Graphische Darstellung der Durchschnittswerte aller CMS bzgl. der genannten Schwachstellentypen 66
Abbildung 3.8: Graphische Darstellung der Schwachstellen in den Erweiterungen und Basis-Installation 67
Abbildung 3.9: Schwachstellen absolut pro CMS 68
Abbildung 4.1: Existierender Benutzername, falsches Passwort 77
Abbildung 4.2: Nicht existierender Benutzername, beliebiges Passwort 77
Abbildung 4.3: Zu Beginn bei leerer Eingabe 78
Abbildung 4.4: Nach Eingabe von PW%pa 78
Abbildung 4.5: Nach Eingabe von PW%passwort+1 78

Tabellenverzeichnis

Tabelle 1.1: Tools für die Erstellung von Websites (ohne HTML-Editoren) 9
Tabelle 2.1: Drupal – Spezifikationsübersicht 14
Tabelle 2.2: Plone – Spezifikationsübersicht 17
Tabelle 2.3: WordPress – Spezifikationsübersicht 20
Tabelle 2.4: Joomla! – Spezifikationsübersicht 23
Tabelle 2.5: TYPO3 – Spezifikationsübersicht 25
Tabelle 2.6: Kriterien – Service Design (Details) 43
Tabelle 2.7: Kriterien – Service Transition (Details) 46
Tabelle 2.8: Kriterien – Service Operation (Details) 49
Tabelle 3.1: Drupal – Meldung aus "Security News“ Archiv 54
Tabelle 3.2: Drupal – Schwachstellenmeldungen 54
Tabelle 3.3: Drupal – CVSS Score 55
Tabelle 3.4: Drupal – Schwachstellen 56
Tabelle 3.5: Plone – Meldung aus "Security News“ Archiv 56
Tabelle 3.6: Plone – Schwachstellenmeldungen 56
Tabelle 3.7: Plone – CVSS Score 57
Tabelle 3.8: Plone – Schwachstellen 58
Tabelle 3.9: WordPress – Meldung aus "Security News“ Archiv 58
Tabelle 3.10: WordPress – Schwachstellenmeldungen 58
Tabelle 3.11: WordPress – CVSS Score 59
Tabelle 3.12: WordPress – Schwachstellen 60
Tabelle 3.13: Joomla! – Meldung aus "Security News“ Archiv 60
Tabelle 3.14: Joomla! – Schwachstellenmeldungen 60
Tabelle 3.15: Joomla! – CVSS Score 61
Tabelle 3.16: TYPO3 – Security Advisories 62
Tabelle 3.17: TYPO3 – Schwachstellenmeldungen 62
Tabelle 3.18: TYPO3 – CVSS Score 63
Tabelle 3.19: TYPO3 – Schwachstellen 64
Tabelle 3.20: Entwicklung der Schwachstellen in verwendeten Basistechnologien 64
Tabelle 3.21: Entwicklung von Schwachstellen Datenbank-Systemen 65
Tabelle 3.22: Durchschnittswerte aller CMS bzgl. der genannten Schwachstellentypen 66

Tabelle 3.23: Schwachstellen in den Erweiterungen und Basis-Installation....67
Tabelle 3.24: Behandlung der Schwachstellen....68
Tabelle 3.25: Schweregrad der Schwachstellen....68

1 Einleitung

1.1 Motivation

Unter einem Content Management System (CMS) wird im Allgemeinen ein System verstanden, das die Verwaltung, die Darstellung und die Nutzung aufbereiteter Informationen (z.B. Texte, Bilder, Grafiken) erleichtert.[1], [2]

Da es sehr unterschiedliche Anwendungsszenarien für die Verwaltung, Darstellung und Nutzung von aufbereiteten Informationen gibt, haben sich im Laufe der Zeit verschiedene Kategorien von CMS entwickelt, die diese Anwendungsszenarien besonders gut unterstützen. Die funktional mächtigste Kategorie ist das Enterprise Content Management System (ECM), welches ein Unternehmen entlang seiner Geschäftsprozesse voll und umfassend unterstützt. Für funktionale Vergleiche von CMS finden sich im Netz einige Portale, die einen ersten Überblick bieten[3].

Diese Studie konzentriert sich allerdings auf Web Content Management Systeme, d.h. CMS, die dafür konstruiert wurden, um die Pflege, Darstellung und Nutzung aufbereiteter Informationen auf einer Intranet- oder Internetsite zu erleichtern. Diese Systeme sind aus sicherheitstechnischer Sicht besonders kritisch, da sie einem Angreifer ein erstes Ziel, eine erste Plattform bieten können, um in eine komplexere Unternehmensstruktur einzudringen. Web Content Management Systeme[4] sind meist Systeme „von der Stange“, die so häufig eingesetzt werden, dass eine bekannt gewordene Sicherheitslücke viele Websites gleichzeitig gefährdet. Bekannte Sicherheitslücken im CMS können von Angreifern leicht dazu missbraucht werden, interne Infrastrukturen ohne großen Aufwand auszuspionieren.

Netzwerktechnisch unterscheiden die Architekten kommunaler Websites oft zwischen einem sicheren Intranet (Behördennetz, Fachanwendungen, Firma etc.), das durch Firewall-Strukturen gesichert ist und einer externen Website, die „völlig getrennt“ von der internen Infrastruktur betrieben wird. Die Hauptaufmerksamkeit der verantwortlichen Stelle liegt häufig bei der internen Struktur.

Die Vernetzung der eigenen Webseite mit externen Internetseiten ist meist von Kunden/Bürgern gewünscht, stellt aber immer ein Sicherheitsrisiko dar. Eine „Entkopplung“ äußerer Darstellung von interner Infrastruktur ist dabei oft keine Option. Dies betrifft nicht nur kommunale Websites mit ihrem Tourismus-, Wirtschafts- und Verwaltungsteil, sondern auch viele Fachverfahren, die sich mittlerweile sehr häufig der Internettechnologien bedienen.

Das Ziel dieses Dokumentes ist es, eine Auswahl von CMS mit dem Schwerpunkt auf Freie Software bezüglich ihrer Sicherheit zu untersuchen und zu bewerten. Die Studie zeigt Privatpersonen und IT-Verantwortlichen der öffentlichen Verwaltung sowie der freien Wirtschaft Handlungsfelder und Gestaltungsoptionen beim Auf- oder Ausbau ihrer Websites mit CMS auf. Dabei muss die rechtliche und organisatorische Ebene ebenso berücksichtigt werden, wie die darunter liegende Technik. Es müssen nicht nur verfügbare Funktionen von CMS verglichen werden, sondern auch der gesamte Lebenszyklus von der Produktauswahl bis zum kontinuierlichen Betrieb des CMS.

1 vgl. Definition von CMS in Wikipedia: http://de.wikipedia.org/wiki/Content-Management-System

2 Für den Begriff „Content Management System“ gibt es fast so viele Definitionen wie Anbieter und Berater. Um die Systeme miteinander vergleichen zu können, werden im Stichwortverzeichnis einige Schlüsselbegriffe definiert.

3 z.B. http://www.cmsmatrix.org/matrix/cms-matrix

4 Um die Lesbarkeit zu vereinfachen wird ab hier nur noch von CMS gesprochen.

1.2 Untersuchungsgegenstand

Der Schwerpunkt dieses Dokumentes liegt auf der technischen Betrachtung des sicheren Einsatzes von CMS.[5] Die einzelnen Phasen des Softwarelebenszyklus werden unter Sicherheitsgesichtspunkten analysiert. Darüber hinaus werden Themen wie beispielsweise Protokollierungseinstellungen und Datenschutz, welche mittelbaren Einfluss auf das Sicherheitsmanagement haben, benannt und wo möglich bewertet.

Die bei CMS Installationen gefundenen Schwachstellen lassen sich neben Konfigurationsfehlern hauptsächlich auf konzeptionelle Mängel bei der Planung der verschiedenen Komponenten und Abläufe zurückführen[6]. Das Dokument wird deshalb einerseits eine grundlegende und verständliche technische Bewertung der Systeme erstellen, andererseits wird es auf die Stolpersteine bei der Einführung der CMS eingehen, um den Verantwortlichen Anhaltspunkte zur Bewertung der Komplexität beim CMS-Einsatz zu geben.

Bei der Behandlung des Themas „Content Management Systeme“ stellt sich die Frage, welche CMS betrachtet werden müssen. Besonders gravierend wirken sich ungepatchte „Monokulturen“[7] in der verwendeten Web-Technologie aus.

In der Tabelle 1.1 wurde die Verteilung der Software[8] dargestellt, mit der Websites in der .de Domain erstellt werden, wobei die Auflistung nur die automatisiert erkennbaren Websites, d.h. etwa 20% der Domains, umfasst. Ein Teil der Einträge verzeichnet reine HTML-Editoren, die hier nicht betrachtet werden. Es zeigt sich, das Open Source Produkte mit professionellem Support wie WordPress, TYPO3 oder Joomla!, weit vorn in der Gunst der Dienstanbieter liegen. Zusätzlich zu den drei meist genutzten Open Source Projekten wurden die Open Source Systeme Drupal und Plone ausgewählt, weil Drupal auf der Weltrangliste Platz vier einnimmt und Plone ebenfalls weit verbreitet ist.

5 Einführende Links zum Thema CMS gibt es im Anhang.

6 Fehlkonfigurationen zählen zu den häufigsten Fehlern, vgl. https://www.owasp.org/index.php/Top_10_2010-A6

7 Fehlerbehaftete Websites, die erkennbar mit ein und demselben Werkzeug erstellt wurden.

8 Dazu gibt es verschiedene öffentliche Statistiken, wie z.B. „Publishing tool statistics for Germany | CMSCrawler.com“; http://www.cmscrawler.com/tld/de; Zugriff am 03.12.2012

Publizierungswerkzeug	*Anzahl der Sites, die das Werkzeug verwenden*	*Marktanteil in %*
WordPress	169442	23.21
TYPO3	129687	17.77
Joomla!	126471	17.32
NetObjects Fusion	43411	5.94
Web2Date	23612	3.23
Contao Open Source CMS	18308	2.51
Adobe GoLive	15673	2.14
Drupal	15594	2.13
xtCommerce	7590	1.04
CMS Contenido	6939	0.95
CM4All	6326	0.86
Incomedia WebSite X5 Evolution	5401	0.74
CMSimple	3281	0.44
CMS made simple	2837	0.38
WEB.DE WebBaukasten	2766	0.37
Plone	2724	0.37

Tabelle 1.1: Tools für die Erstellung von Websites (ohne HTML-Editoren)[9]

1.3 Untersuchungsmethodik

Zunächst muss über den typischen Projektzeitraum einer CMS-Einführung von der Produktauswahl bis zur Außerbetriebnahme untersucht werden, welche Eigenschaften oder Rahmenbedingungen sicherheitsrelevant sein können. Hier gibt es klare Abstufungen: So wirken sich beispielsweise gemeldete Schwachstellen, die nicht zeitnah behoben werden, direkt negativ auf die Empfehlung eines CMS aus, während eine schlechte Integrationsfähigkeit in vorhandene Systemmanagementumgebungen dagegen eher mittelbar Einfluss auf die Sicherheitseigenschaften hat.

Die Kriterien, die zur Untersuchung der einzelnen Systeme verwendet werden, werden zu Beginn aufgestellt und erklärt. Der Schwerpunkt der Untersuchung liegt dabei klar auf der Architektur und auf dem konzeptionellen Aufbau der Systeme. Die technische Struktur der Komponenten, ihre Kommunikationsbeziehungen und ihre gegenseitigen Absicherungsmöglichkeiten stehen im Vordergrund, wo möglich wird auch die interne Softwarearchitektur untersucht.

Für die eingehende Beurteilung der IT-Sicherheit ist an vielen Stellen ein Schwachstellentest empfehlenswert. Da es sich bei dieser Studie um eine erste Einschätzung und Basisuntersuchung der CMS handelt, wird auf die Durchführung solcher Test verzichtet. Zudem könnte ein Test einer speziell konfigurierten Umgebung dazu verleiten, das verwendete CMS generell als „sicher“ zu betrachten, ohne einen individuellen Test durchzuführen[10]. Die Einschätzung der Systeme erfolgt überwiegend auf Basis der Dokumentenlage, die auszugsweise durch eigene Installationen und funktionale Tests verifiziert wird. Um zu verdeutlichen, welche Kriterien im Rahmen eines sicheren CMS Betriebs abgeprüft werden müssen, werden die Kriterien benannt und als „nicht getestet“ markiert.

Häufig werden in Untersuchungen die mitgelieferten Standard-Installationen bewertet. Dagegen spricht, dass die „System-Härtung“ traditionell nach der Erstinstallation stattfindet. Viele Systeme werden mit Sicherheitslücken ausgeliefert, die der professionelle Administrator kennt und nach der Installation

9 Quelle: „Crawler Statistics – metagenerator.info“; http://www.metagenerator.info/statistic.html; Zugriff am 06.09.12

10 Davon ist dringend abzuraten, da die Anzahl der zu beachtenden Parameter sehr hoch ist.

schließt. Um Vergleichbarkeit zwischen den Systemen zu erreichen, werden die Installationen durch professionelle Administratoren begleitet und es wird der Umfang der jeweils notwendigen Arbeiten bewertet.

Man muss bedenken, dass ein System, das heute als sicher eingestuft wird, in naher Zukunft ggf. gravierende Sicherheitsrisiken aufweisen könnte. Beispielsweise kann jedes neue Release Sicherheitslücken mit sich bringen, die erst im Prozess der weltweiten Nutzung bemerkt werden. Schon die Auswahl der untersuchten Version kann also unterschiedliche Ergebnisse liefern. Da die Auslieferungspolitik immer auch ein Untersuchungskriterium ist, betrachtet diese Studie jeweils immer die aktuellsten, als stabil bezeichneten Versionen.

Die IT-Sicherheit wird wesentlich von rechtlichen und organisatorischen Rahmenbedingungen beeinflusst. Der Redaktionsworkflow ist ein deutliches Beispiel dafür. Da dies eine generische Betrachtung der CMS sein soll, werden die Systeme danach beurteilt, wie sie organisatorische „Best Practices“, z.B. das Vier-Augen-Prinzip, unterstützen. Das organisatorische Umfeld wird einbezogen, in dem domänenspezifische Szenarien[11] gebildet werden. Auch auf Basis dieser Szenarien wird die Eignung der CMS eingeschätzt.

Auch der Support des CMS Herstellers bzw. der Dienstleister, die für das ausgewählte CMS Leistungen anbieten, ist – beginnend von der Planungsphase bis zum Betrieb – wichtig für eine sichere CMS Nutzung. Um hier eine Einschätzung zu erhalten, können stichprobenartige Interviews mit Kunden geführt werden, in denen ermittelt wird, inwieweit der Support IT-Sicherheitsaspekte behandelt und in welcher Qualität die Sicherheitsfragen beantwortet wurden. Bei den Open Source Projekten ist die Qualität der Security Foren/Newsticker transparent und kann direkt bewertet werden.

1.4 Aufbau der Studie

In Kapitel 2 werden die Grundlagen für die Sicherheitsuntersuchungen gelegt: Aufbauend auf der technischen Architektur eines Muster-CMS, die am Anfang des Kapitel 2 erläutert wird, werden die ausgewählten CMS mit diesem Muster verglichen. Um den Vergleich der CMS untereinander und ihre Bewertung vornehmen zu können, werden Bewertungskriterien aufgestellt und kurz erläutert. Den Abschluss von Kapitel 2 bildet eine Reihe typischer Szenarien, die den funktionalen Umfang der im Anschluss geprüften CMS eingrenzen.

Kapitel 3 enthält eine dokumentenbasierte Analyse der aktuellen Bedrohungslage mit den bemerkten Schwachstellen.

Kapitel 4 bewertet die CMS auf Basis der oben entwickelten Kriterien hinsichtlich ihrer Sicherheitseigenschaften.

Kapitel 5 führt alle Ergebnisse zusammen und bewertet die Eignung der Systeme für die im Kapitel 2 beschriebenen Szenarien.

1.5 Notation

Um ein einheitliches Vorgehens bei der Erstellung der Studie über verschiedene Teildokumente und Phasen zu wahren, wird prinzipiell unterschieden zwischen:

- Fakten, die mit Dokumentationen und Beobachtungen belegbar sind und
- Empfehlungen zu kontinuierlichen oder einmaligen Maßnahmen

Zur besseren Lesbarkeit sind diese unterschiedlichen Textteile farblich abgehoben und werden abschließend im Anhang zusammengefasst.

11 vgl. Kapitel 2.4

Fakten werden optisch grün und durch das vorangestellte Wort „FAKT“ gekennzeichnet.

Empfehlungen (mittel- bis langfristige Maßnahmen) werden optisch rot sowie durch das vorangestellte Wort „EMPFEHLUNG“ gekennzeichnet.

2 Einführung in Content Management Systeme (CMS)

2.1 Technischer und funktionaler Aufbau

Websites, insbesondere Informationsportale im Internet, leben von verlässlicher Information, qualitativ hochwertigen Inhalten und zeitnaher Aktualisierung. Mit CMS können Dienstanbieter die Benutzer schnell, kostengünstig und zuverlässig mit den im Kontext relevanten Informationen versorgen. Mit zunehmender Komplexität der Websites werden CMS als technische Basis unabdingbar, um die Vielzahl an Inhalten und Formaten zu pflegen und zu organisieren. Die zentrale und formatunabhängige Datenhaltung ermöglicht es, Inhalte auch mehrfach und in verschiedenen Internetangeboten zu publizieren. Häufig werden CMS um E-Commerce-, E-Participation- oder E-Collaboration-Lösungen erweitert.

Der technische Aufbau der zu betrachtenden Systeme ist schematisch in Abbildung 2.1 dargestellt. Die Systeme unterscheiden sich sowohl hinsichtlich der Basis-Technologien als auch im Hinblick auf die Abgrenzung der logischen Komponenten. Grundsätzlich folgen CMS den Prinzipien moderner Software-Architekturen.

Es gibt einen Kern und darauf aufbauende Erweiterungen, über die alle nicht im Kern enthaltenen Funktionen abgebildet werden. Erweiterungen können dabei vom Hersteller des Systems oder auch von externen Entwicklern zur Verfügung gestellt werden. Das System nutzt Bibliotheken des Betriebssystems für Basisfunktionalitäten.

Die zentrale Kernkomponente jedes CMS ist die Inhaltsverwaltung, einschließlich der daran anknüpfenden Dienste. Zu diesen Diensten gehört auch die Transformation der Inhalte auf verschiedene Ausgabeformate. Damit Inhalte auf PC, Tablet und Smartphone dargestellt werden können, werden die Inhalte mit Metadaten versehen, die vor der Ausgabe ausgewertet werden. Auch der Mechanismus zum Entfernen von Informationen nach Ablauf von Fristen (Retention), die Internationalisierung von Inhalten (verschiedene Sprachversionen) und die zentrale Suchmaschine greifen auf die Metadaten zurück.

Über einen Mechanismus zur Aufbereitung von Inhalten anhand von Vorlagen, häufig als Template Engine bezeichnet, können Inhalte mit Vorlagen verknüpft und für die Darstellung vorbereitet werden. Die Nutzung einer einheitlichen Vorlage erleichtert z.B. die Umsetzung spezieller Anforderungen wie die barrierefreie, BITV[12] konforme Darstellung von Webseiten.

Über die Nutzerverwaltung des CMS lassen sich Nutzer und deren Rechte zum Zugriff auf Ressourcen wie z.B. geschützte Bereiche der Außensichten einstellen. Bei einigen CMS werden Prinzipien der Vorgangsbearbeitung wie aufeinander folgende Aufgaben und Freigaben mit Mehr-Augen-Prinzip anhand sogenannter Workflows unterstützt.

12 BITV – Barrierefreie Informationstechnik-Verordnung; http://www.barrierefreies-webdesign.de/bitv/

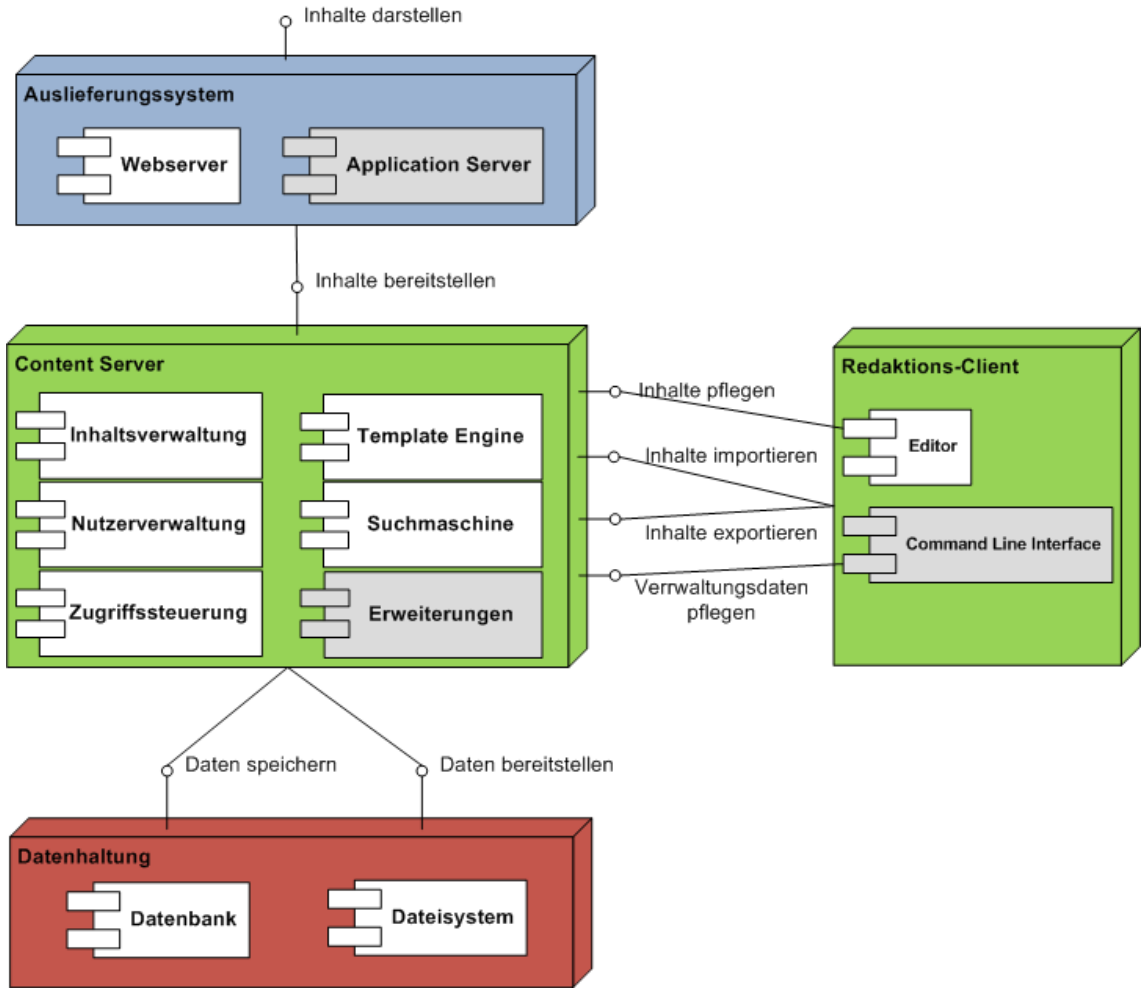


Abbildung 2.1: Generisches UML Deployment Diagramm

Die Komponenten Inhaltsverwaltung, Nutzerverwaltung, Zugriffssteuerung, Template Engine und Suchmaschine sind im Content Server (vgl. Abbildung 2.1 Mitte) zusammengefasst. Der Zugriff auf den Content Server erfolgt über einen Redaktions-Client. Redaktions-Client und Content Server kommunizieren über ein standardisiertes oder proprietäres Protokoll zur Erstellung und Pflege von Inhalten, aber auch zur Umsetzung des Redaktionsworkflows.

Der Redaktions-Client enthält als wichtigste Komponente einen Editor, über den die Redakteure Inhalte und deren Metadaten erfassen und pflegen. Zusätzlich steht bei einigen Systemen ein Command Line Interface (CLI) zur Verfügung.

Die Datenhaltung der CMS-Inhalte erfolgt separat von der eigentlichen CMS-Kerntechnologie. In allen CMS dieser Studie wird eine Datenbank zur Speicherung der Inhalte und Metadaten verwendet. Zusätzlich wird das Dateisystem zur Ablage von Konfigurationsdaten und Inhalten genutzt.

Das Auslieferungssystem besteht mindestens aus einem Webserver zur Darstellung der Inhalte. Neben dem Webserver kommt in umfangreicheren CMS ein Application Server zum Einsatz. Dieser Server stellt zusätzliche Extras, wie die Erstellung dynamischer Inhalte oder die Kommunikation über ein Webformular zur Verfügung.

Die Ein- und Ausgabe der im CMS verwalteten Inhalte kann über verschiedene Kanäle erfolgen. So können neben der Darstellung als Webseite auch Exporte im XML Format oder auch Newsfeeds realisiert werden. Der Import von Inhalten kann ebenso über verschiedene Formate erfolgen. Letztlich kann auch der Editor als Eingabekanal bezeichnet werden.

2.2 Beschreibung der ausgewählten Systeme

2.2.1 Drupal

Spezifikation	*Drupal 7.17*
Betriebssystem	Windows, Unix, Linux
Eingesetzte Programmiersprache	PHP 5.3 oder höher
Webserver / Applikationscontainer	Apache, Nginx, Microsoft IIS
Datenhaltung / Datenbank	MySQL 5.0.15 oder höher mit PDO, PostgreSQL 8.3 oder höher mit PDO, SQLite 3.3.7 oder höher, MariaDB 5.1.44 oder höher, Microsoft SQL Server und Oracle werden unterstützt, wobei viele Module für MySQL geschrieben und nicht unter kommerziellen Datenbanken getestet wurden
Redaktions-Client	Alle gängigen Browser, die CSS unterstützen
Lizensierung	Open Source unter GPLv2 oder später

Tabelle 2.1: Drupal – Spezifikationsübersicht

2.2.1.1 Systemarchitektur

Das System basiert auf der LAMP-Architektur.[13] Wie in Abbildung 2.2 dargestellt bilden das Auslieferungssystem und der Content Server ein System.

Die typische PHP-Instanz ist nicht über mehrere Server verteilbar. Somit sind externe Lastverteilungsmechanismen notwendig. Es existiert eine Reihe von Caching-Modulen, die für verschiedene Anwendungsszenarien genutzt werden.

FAKT: Lastverteilung wird bei allen PHP-basierten CMS über externe Mittel realisiert.

Das Zusammenstellen einer PHP-basierten hochperformanten Site[14] für viele Benutzer ist mindestens so anspruchsvoll wie mit Java EE Technologie. Deshalb gibt es relativ wenige Firmen, welche die dazu notwendige Expertise besitzen.

Die Integration in serviceorientierte Umgebungen ist möglich, sowohl als Web Service Client als auch als Server, allerdings sind die für .net oder Java vorhandenen Implementierungen von Standards zur Sicherstellung von Integrität und Vertraulichkeit auf Nachrichtenebene – WS-Security[15] etc.– nicht vorhanden.

FAKT: WS-Security wird von keinem der CMS unterstützt.

13 LAMP – Linux, Apache HTTP, MySQL und PHP
14 vgl. The Economist: http://www.economist.com/
15 https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

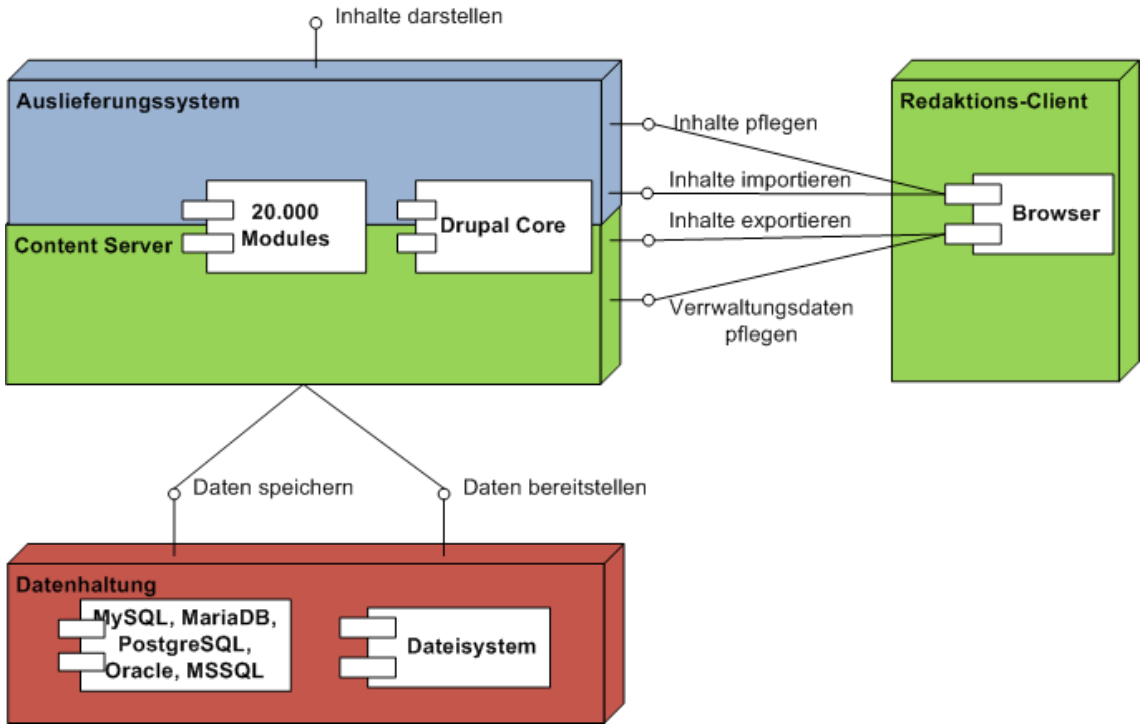


Abbildung 2.2: Drupal – Systemarchitektur

2.2.1.2 Komponenten

Vom Ansatz her ist Drupal als ein Content-Management-Baukasten[16] mit wohldefinierten Kernkomponenten zu verstehen. Drupal besteht aus einem minimalen Kern mit tausenden Erweiterungen (modules), die durch aktuell 512 Distributionen[17] vorkonfiguriert sind. Die Einstellung der Module ist meist in zwei Aspekte untergliedert: Rechtevergabe und Konfiguration. Konfiguration ist generell einfach. Die Erstellung von Inhaltsseiten erfordern tiefere Kenntnisse in Drupal. Core Module sind nach kurzer Einarbeitungsphase in die Struktur von Drupal leicht verständlich. Erweiterungsmodule hingegen sind in ihrer Konfiguration meist unübersichtlich und erfordern tiefere Kenntnisse in Drupal.

FAKT: Die Erstellung einer Website mit Drupal erfordert die Einarbeitung in die Basistechnologien, was über rein konfigurative oder administrative Tätigkeiten deutlich hinausgeht.

Die zentralen funktionalen Aspekte Inhalts- und Nutzerverwaltung, Zugriffssteuerung und Template Engine sind Teil des Kerns. Die Anbindung von Suchmaschinen ebenso wie die Administration per Kommandozeile, insbesondere von remote Drupal Instanzen, wird über Erweiterungen (vgl. drush[18]) abgebildet. Als WYSIWYG Editor lässt sich der meist verwendete TinyMCE[19] über eine Erweiterung einbinden.

Teil des Kerns ist ein System zur Eingabefilterung. Filter können je nach Vertrauenswürdigkeit bestimmten Benutzergruppen zugeordnet werden. So kann beispielsweise anonymen Benutzern verboten werden, den Anchor-Tag zu verwenden, während voll vertrauenswürdige Nutzer selbst PHP Code oder Javascript einbetten dürfen.

FAKT: Es gibt bei Drupal prinzipiell ein Recht, ausführbaren Code oder Javascript einzubetten.

16 Es wird explizit auf das „Lego“ Prinzip verwiesen.
17 Zusammenstellungen von Modulen
18 http://drupal.org/project/drush
19 http://www.tinymce.com/

Die Aktualisierung bzw. Erweiterung des Drupal Systems erfolgt per Kommandozeile. Das System informiert den Administrator, dass Sicherheitspatches vorliegen, die eingespielt werden müssen.

2.2.1.3 Protokolle, Kommunikation und Datenflüsse

Die Administration der Site erfolgt vom Server lokal über die Drupal-Shell „drush“ oder über einen Web-Browser. Sobald über drush remote drupal Instanzen verwaltet werden sollen, können die Verbindungen über SSH getunnelt werden.

FAKT: Drupal ermöglicht über drush die Verwaltung mehrerer Websites auf sichere Art und Weise (SSH-Tunnel).

Alle redaktionellen Tätigkeiten zwischen Redaktions-Client und Content Server erfolgen per Web-Browser über http bzw. https.

Die Kommunikation zwischen Drupal und MySQL erfolgt über einen lokalen Socket oder über TCP (standardmäßig auf Port 3306). Die MySQL-Authentifizierung ist so gestaltet, dass ein Angreifer aus dem Abhören der Kommunikation nicht auf das Passwort schließen kann. Eine etablierte Session versendet Daten im Klartext, beim Eröffnen der Session kann jedoch angegeben werden, dass SSL verwendet werden soll.

FAKT: Die Kommunikation der php-basierten Systeme zur mysql Datenbank erfolgt nach dem Stand der Technik.

2.2.1.4 Funktionalitäten

Drupal hat sehr limitierte Möglichkeiten zum Umsetzen eines Redaktionsworkflows. Inhalte werden direkt, ähnlich wie bei einem Wiki, durch das Publizieren geändert. Durch vorhandene Erweiterungsmodule ist es möglich, diesen Mechanismus durch Zwischenzustände, d.h. Positionen zum Freigeben von Inhalten, zu ergänzen.

Der Seitenerzeugung ähnelt im Aufbau einem Portal-Server. Die Autorisierung geht bis auf Attribut-Ebene, d.h. es kann mittels PHP beispielsweise gesteuert werden, dass angemeldete Benutzer in einer „Region“ der Seite verschiedene Inhalte, sogenannte „Blöcke“, angezeigt bekommen. Es kann auch erlaubt werden, dass der Benutzer selbst einen Block wiederum nicht anzeigt. Dadurch ergibt sich ein hoch flexibles aber auch schwer testbares Framework.

2.2.1.5 Besonderheiten

Die Drupal Community kann als eine der derzeit aktivsten Open Source Communities bezeichnet werden. Das Drupal Projekt legt sehr viel Wert auf sichere Codierung[20]. Es gibt ein Peer Review (Patchbasierte Entwicklung mit Mehr-Augen-Prinzip) und ein dezidiertes Security Team[21] mit definiertem Workflow, Reporting/Support und Disclosure Policy.

Die Anforderungen an die Administration der Site sind verglichen mit WordPress oder Joomla! deutlich höher. Das Konfigurationsmanagement ist anspruchsvoll.

20 vgl. Code Standards http://drupal.org/coding-standards und http://drupal.org/writing-secure-code
21 http://drupal.org/security-team

2.2.2 Plone

Spezifikation	*Plone 4.2.1*
Betriebssystem	Windows, Unix, Linux, BSD, OS X
Eingesetzte Programmiersprache	Python 2.6 oder höher
Webserver / Applikationscontainer	Zope
Datenhaltung / Datenbank	Zope Object Database (ZODB), Zugriff auf SQL DBs über Database Adapter möglich
Redaktions-Client	Alle gängigen Browser, die CSS unterstützen
Lizensierung	Open Source unter GPLv2

Tabelle 2.2: Plone – Spezifikationsübersicht

2.2.2.1 Systemarchitektur

Im Unterschied zu den PHP Systemen läuft Plone in einem Applikationsserver (Zope[22]) und weist deshalb die Vorzüge der Lastverteilung auf Anwendungsebene auf. Im Unterschied zur Absicherung der Website durch systemnahe Dienste (z.B. Cluster aus Linux Servern) kann ein Benutzer seinen persönlichen Vorgang selbst dann fortsetzen, wenn der Zope Server, auf dem er gerade gearbeitet hat, ausfällt. Dies ist natürlich auch für alle Websites mit hohen Lastanforderungen[23] interessant. Zope Enterprise Objects (ZEO) ist der empfohlene Weg, um den aufwändigen Rendering-Prozess auf endlich viele Clients zu verteilen. Alle Clients, die über einen Loadbalancer[24] zu einem Browser zugeordnet werden müssen, verbinden sich an einen ZEO-Storage Server. Bei den ZEO-Clients handelt es sich intern auch um vollwertige Zope-Applikationsserver.

FAKT: Die Lastverteilung funktioniert bei Plone über den Zope Applikationsserver.

Für die Absicherung des ZEO-Storage Servers können Mechanismen und Komponenten aus dem Linux-HA Projekt[25] verwendet werden. Es gibt zudem einige kommerzielle Angebote von IT-Dienstleistern dazu.

2.2.2.2 Komponenten

Die zentralen funktionalen Aspekte des Content Management Frameworks (CMF) sind die Inhalts- und Nutzerverwaltung, die Zugriffssteuerung, die Template Engine, die Suchmaschine ebenso wie der Redaktionsworkflow und die Unterstützung verschiedener Content Types. Mit der Installation des Kerns ist deshalb schon eine voll funktionsfähige Website vorhanden.

Die Installation weiterer Module aus einem gepflegten Repository auf plone.org erfolgt über ein Installationsprogramm auf der Kommandozeile. Danach lassen sich die installierten Module über die Website aktivieren oder deaktivieren.

22 http://www.zope.de/

23 Hohe Lastanforderungen sollten im Einzelfall diagnostiziert werden. Um ein Maß zu haben, wird dies hier trotzdem für die Allgemeinheit versucht: Wenn mehr als 500 Benutzer gleichzeitig auf einer Webseite arbeiten und davon mehr als 10% der Benutzer Daten verändern, dann wird dies als hohe Lastanforderung betrachtet.

24 Lastenverteiler; vgl. http://de.wikipedia.org/wiki/Lastverteilung_%28Informatik%29

25 http://www.linux-ha.org/wiki/Main_Page

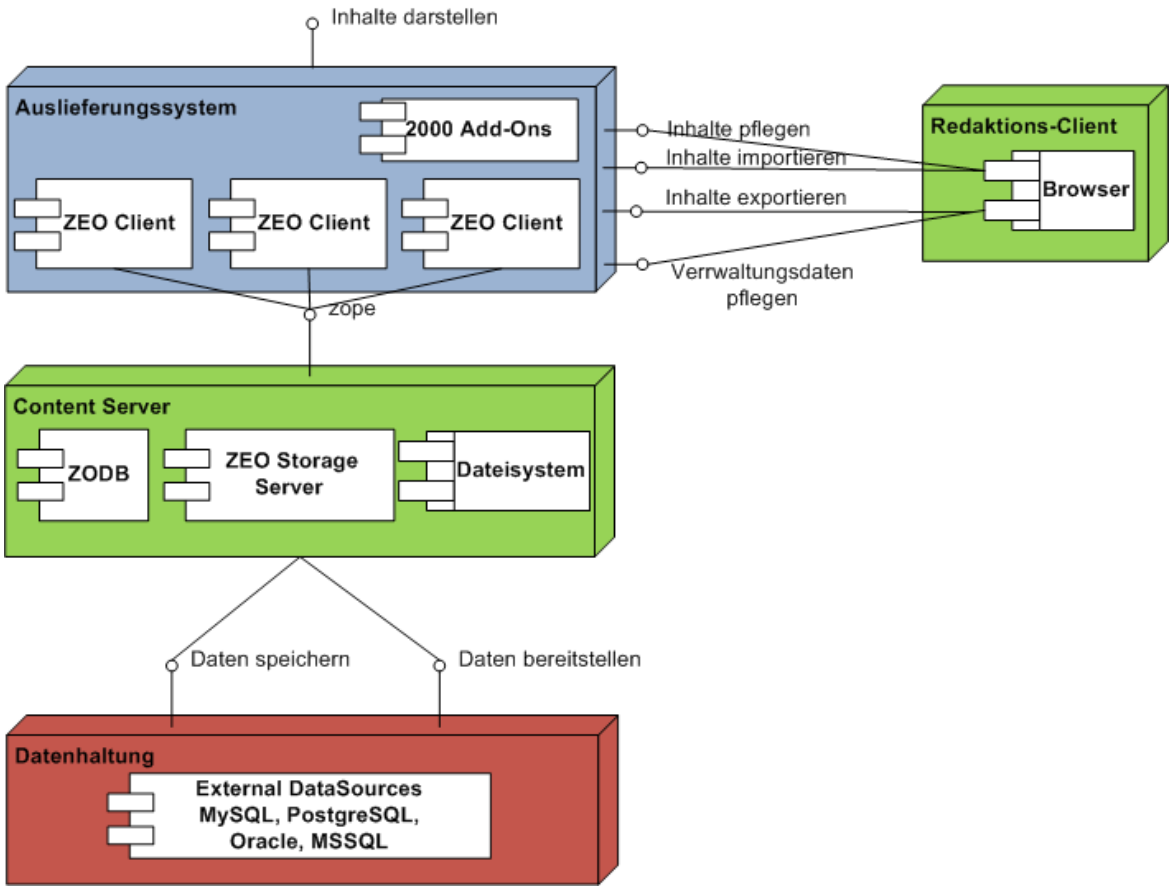


Abbildung 2.3: Plone – Systemarchitektur

Funktional enthält das Plone-Repository eine Auswahl von über 2000 Add-Ons. Die Plone Standard-installation beinhaltet schon 15 Module, die über die Website aktiviert oder deaktiviert werden können. Die Funktionalität dieser Module variiert von einem kompletten browserunabhängigen visuellen Editor (Kupu[26]) bis zu einem Modul zur Unterstützung des Cachings von HTTP. Die Standardinstallation ist ohne weitere Module bereits geeignet, eine kleine Website zum Austausch von Informationen und Dokumenten aufzusetzen.

2.2.2.3 Protokolle, Kommunikation und Datenflüsse

Die Kommunikation zwischen ZEO-Clients und Server läuft über ein eigenes Protokoll, welches entfernte Prozeduraufrufe realisiert. Diese Kommunikation wird in der Praxis über SSH getunnelt, um die Authentizität der Kommunikationspartner und die Vertraulichkeit und Integrität der transferierten Daten zu sichern.

FAKT: Die Komponenten des Plone-CMS kommunizieren über entfernte Prozeduraufrufe, die über SSH getunnelt werden können.

2.2.2.4 Funktionalitäten

Plone ermöglicht verschiedenste Workflows, Content kann mehrere Zustände (z.B. privat, öffentlicher Entwurf, zur Redaktion eingereicht, veröffentlicht) annehmen. In jedem Zustand gibt es eine Versionierung,

26 http://plone.org/products/kupu/

so dass ein Redakteur immer den letzten Stand wiederherstellen kann. An die Zustandsübergänge können Funktionalitäten (z.B. E-Mailversand) geknüpft werden. Es gibt mehrere Workflowtypen, die bereits in der Standardinstallation enthalten sind.

Die redaktionelle Arbeit findet nicht über den Content Server, sondern auch über das Auslieferungssystem statt. Es ist aber sehr gut möglich, sämtliche administrative Tätigkeiten nur über einen dedizierten ZEO-Client abzuwickeln, der dann speziell abgesichert werden kann.

FAKT: Die Administration der Plone Site kann über einen dedizierten Client speziell abgesichert werden.

Es existiert eine detailreiche Autorisierung. Rechte sind ausschließlich an Rollen geknüpft wobei hier zwischen globalen und lokalen Rollen unterschieden wird. Globale Rollen beziehen sich auf die gesamte Site, lokale Rollen sind kontextabhängig und können sich auf Elemente einer Site beziehen. Die Redaktionsworkflows werden über Richtlinien bestimmt und können sehr weit konfiguriert werden. Die Auslieferung der Informationen kann stark optimiert (gepackt) werden.

2.2.2.5 Besonderheiten

Dem Administrator stehen viele in Python geschriebene Werkzeuge zur Verfügung, mit denen alltägliche Aufgaben erleichtert werden können.[27]

Als Nachteil gegenüber anderen Systemen fällt die „Exotik“ der Zope Object Database (ZODB)[28] auf. Der Dienstanbieter ist auf die Werkzeuge und Reportingmöglichkeiten der ZODB angewiesen und hat auf Grund fehlender Interoperabilität auch keine Chance das Reporting über alternative Werkzeuge abzubilden.[29]

FAKT: Plone verwendet eine eigene, objektorientierte Datenbank.

Plone zählt 340 Kern-Entwickler, d.h. deutlich weniger als z.B. Drupal. Da Python als Programmiersprache im Serverumfeld sehr verbreitet ist, gilt die Sprache als überaus reif. Eine der grundlegenden Prämissen von Python „no surprise“ wirkt sich für den Dienstleister im Betrieb sehr positiv aus. Plone ist vergleichbar wartungsarm. Das System wirbt damit, den besten Track Record von Schwachstellen (CVSS)[30] zu haben, was im Kapitel 3.2.2 genauer untersucht wird. Es gibt Lösungsanbieter überall auf der Welt.

2.2.3 WordPress

WordPress wurde als „personal publishing system" als Nachfolger von „b2/cafelog" entwickelt. Ursprünglich als Werkzeug zur Erstellung von Web-Blogs gedacht, lassen sich durch die hohe Anpassbarkeit auch CMS Funktionalitäten abbilden und einsetzen.

27 Beispiele: Testen von Websites: http://twill.idyll.org; Testen von Netzwerken: http://www.secdev.org/projects/scapy/

28 http://zodb.org/

29 Selbst bei Nutzung der Anbindung an relationale Datenbanken, liegen die Daten als objektrelationale Abbildung vor und sind damit nicht gut auswertbar.

30 http://en.wikipedia.org/wiki/CVSS

Spezifikation	*WordPress 3.4.2*
Betriebssystem	Windows, Unix, Linux
Eingesetzte Programmiersprache	PHP 5.4.2 oder höher
Webserver / Applikationscontainer	Apache oder Nginx wird empfohlen
Datenhaltung / Datenbank	MySQL 5.0 oder höher
Redaktions-Client	Browser bzw. Editoren als Apps für IOS, Android, BlackBerry, Windows Phone, Nokia und für HP WebOS
Lizensierung	Open Source unter GPLv2

Tabelle 2.3: WordPress – Spezifikationsübersicht

2.2.3.1 Systemarchitektur

WordPress basiert auf der LAMP-Architektur.[31] Wie in Abbildung 2.4 dargestellt bilden das Auslieferungssystem und der Content Server ein System. WordPress setzt keinerlei Hochverfügbarkeits- oder Cachingkonzepte um. Stattdessen wird ein schlankes System zur Verfügung gestellt, das sich auf die Kernaufgabe, d.h. die Realisierung eines Blogsystems, konzentriert. Große WordPress Websites verwenden eine Schicht aus Lastverteilern, z.B. auf Basis von pound[32] und wackamole[33], eine weitere Schicht mit den WordPress Auslieferungs-/Contentsystemen, die im Cluster arbeiten, und eine dritte Schicht mit mehreren MySQL Servern im Master/Slave Modus als Datenhaltung. Alternativ existieren jedoch auch Plugins, die das Caching innerhalb der WordPress Auslieferungssysteme realisieren.

FAKT: Die konsequente Reduktion des Systems WordPress auf wesentliche Blogging-Funktionalität ist das herausragende Merkmal gegenüber allen anderen hier betrachteten Systemen.

Zu den Best Practices von WordPress Websites gehört es, „zustandslos“ zu arbeiten, d.h. jeder der Auslieferungsserver im Cluster kann jeden eingehenden Request bearbeiten, weil er die Vorgeschichte der letzten Requests des Benutzers nicht kennen muss. Deshalb ist keine Session-Replikation notwendig und es sind mit geringen Hardwareanforderungen sehr viele Benutzer gleichzeitig zu bedienen.

Sobald WordPress Systeme jedoch durch zusätzliche Module um Funktionalitäten erweitert werden, die sie funktional gleichwertig zu den anderen hier betrachteten CMS machen, unterliegen sie bzgl. des Lastverhaltens sowie der Hochverfügbarkeit ähnlichen Restriktionen wie die anderen PHP basierten CMS.

31 LAMP – Linux, Apache HTTP, MySQL und PHP
32 http://www.apsis.ch/pound/
33 http://www.cnds.jhu.edu/download/download_wackamole.cgi

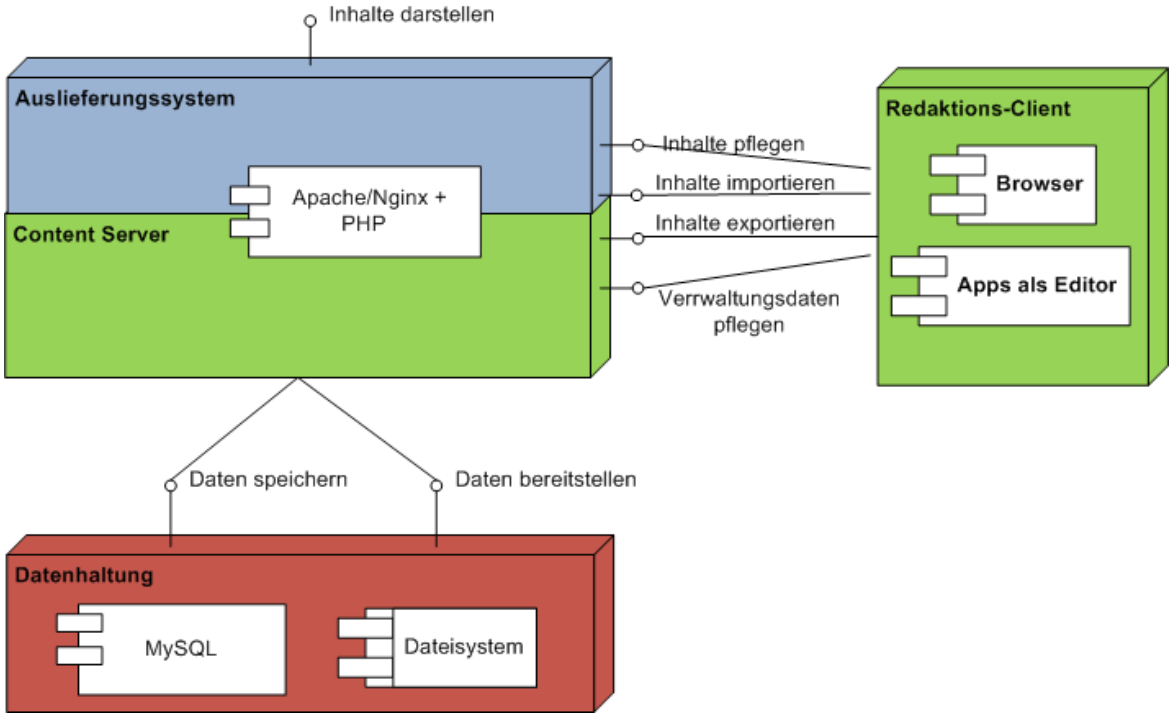


Abbildung 2.4: WordPress – Systemarchitektur

WordPress zeichnet sich durch die Erweiterbarkeit mit über 20.000 Modulen von Drittanbietern aus, welche die Funktionalität von WordPress umfassend auch in Richtung CMS erweitern. Oft werden Funktionalitäten im Kern, die ursprünglich für das Blogging konzipiert waren, für CMS Anforderungen ausgebaut, z.B. werden Redaktionsworkflows realisiert, indem Tags auf den Seiten als Freigabestatus genutzt werden. Natürlich besteht in der Einbindung der Module von Fremdanbietern auch die Gefahr, deren Sicherheitslücken in die eigene Installation aufzunehmen und damit das ganze System anfällig zu machen.

Die Aktualisierung bzw. Erweiterung des WordPress Systems erfolgt teilweise per Browser und teilweise per Kommandozeile. Das System informiert den Administrator, dass Sicherheitspatches vorliegen, die eingespielt werden müssen. Der Update-Prozess erfordert einige wenige Backup- und Konfigurationsschritte, die klar strukturiert sind, jedoch einen völlig unbedarften Administrator leicht überfordern. Dies steht im Widerspruch dazu, dass das System mit einer sehr einfachen, fünfminütigen (Erst-)Installation wirbt.

2.2.3.2 Protokolle, Kommunikation und Datenflüsse

Alle administrativen und redaktionellen Tätigkeiten zwischen Redaktions-Client und Content Server erfolgen per Web-Browser über http bzw. https.

Die Kommunikation zwischen WordPress und MySQL erfolgt über einen lokalen Socket oder über TCP (standardmäßig auf Port 3306), wie bereits bei Drupal beschrieben.

2.2.3.3 Funktionalitäten

Standardmäßig unterstützt WordPress Funktionalitäten, die für das Schreiben von Blogs notwendig sind. Im Bereich Template Engine gibt es deshalb kaum Abstriche gegenüber ausgereiften CMS. Auch TinyMCE[34], ein guter Editor, der bereits Preview- und Drag-and-Drop Funktionen enthält, gehört zum Kern des Systems. Gleichfalls wird standardmäßig das Plugin Akismet[35] installiert, welches vor SPAM schützt.

34 Open Source HTML WYSIWYG Editor von Moxiecode Systems, AB
35 http://akismet.com/

Im Bereich Nutzer- und Zugriffsverwaltung muss man jedoch auf externe Module zurückgreifen.

Die Inhaltsverwaltung ist naturgemäß auf Texte beschränkt und muss über vorhandene Erweiterungsmodule ergänzt werden.

Da Blogs im Unterschied zu modernen Websites keine Inhalte aus verschiedenen Quellen aufnehmen (z.B. Wetterinformationen, Agenturmeldungen etc. zusätzlich zu eigenen Inhalten des Betreibers)[36] ist auch die Suchmaschine nur über externe Module einbindbar. Der Ansatz eines Systems zur Realisierung von Blogs ist es eher, von extern vorhandenen Suchmaschinen gefunden zu werden. Deshalb gibt es sehr viele Erweiterungen zur Optimierung der Auffindbarkeit.[37]

2.2.3.4 Besonderheiten

WordPress hatte in der Vergangenheit häufige, schnell aufeinander folgende Update-Zyklen im Rahmen von funktionalen Patches. Ein kontinuierlicher und konsistenter Update-Prozess wurde damit für Betreiber erschwert, da dieser allein durch die Häufigkeit, zeit- und damit kostenintensiv war.

Auf der Webseite von WordPress[38] wird schon bei den Anforderungen darauf hingewiesen, das WordPress als PHP-Applikation mit einem unprivilegierten Systemnutzer zur Laufzeit betrieben werden sollte. Als Werkzeug wird auf suPHP[39] verwiesen.

2.2.4 Joomla!

Joomla! ist aus dem Open Source-Projekt mambo hervorgegangen und hat ebenfalls eine sehr lebhafte Community, die viele Erweiterungen erstellt. Es ist sehr einfach zu installieren und hat eine sehr ansprechende Benutzeroberfläche, so dass auch ungeübte Benutzer gut mit der CMS Funktionalität klarkommen können.

FAKT: Joomla! verfügt über eine Administrationsoberfläche, die auch für ungeübte Benutzer verständlich und nutzbar ist.

36 vgl. Content Syndication
37 vgl. Search Engine Optimization (SEO)
38 http://wordpress.org/about/requirements/
39 http://www.suphp.org

Spezifikation	*Joomla! 3.02*
Betriebssystem	Windows, Unix, Linux
Eingesetzte Programmiersprache	PHP 5.3.1+ empfohlen
Webserver / Applikationscontainer	Apache 2.x+ mit mod_mysql, mod_xml, and mod_zlib , Nginx 1.1, Microsoft IIS 7
Datenhaltung / Datenbank	MySQL 5.1+ mit PDO, PostgreSQL 8.3.18+ oder höher mit PDO, Microsoft SQL Server 10.50.1600.1+
Redaktions-Client	Alle gängigen Browser, die CSS unterstützen
Lizensierung	Open Source unter GPLv2 oder später

Tabelle 2.4: Joomla! – Spezifikationsübersicht

2.2.4.1 Systemarchitektur

Ähnlich wie bei WordPress ist der Content Server nicht vom Auslieferungssystem getrennt, (vgl. Abbildung 2.5), wodurch die Joomla! Instanz zum Single Point of Failure[40] wird. Darüber hinaus ergeben sich – ähnlich wie bei WordPress – Skalierungsprobleme bzw. Optimierungsmöglichkeiten.

Die Softwarearchitektur des CMS gliedert sich in die Joomla!-Plattform für Web- und Kommandozeilen-anwendungen und in die CMS Applikation auf Basis der Plattform.

FAKT: Im Unterschied zu WordPress ist Joomla! für Erweiterbarkeit konzipiert, was man an den bereits im Kern vorgesehenen Komponententypen erkennen kann.

2.2.4.2 Komponenten

Die Joomla! Sprachwelt unterscheidet die Komponententypen: Components[41], Modules[42] und Plugins[43]. Eine festgelegte Menge dieser Komponententypen bildet die Core-Features. Daneben werden Templates[44] und Languages genannt, die wie in den anderen CMS für das Seitenlayout und die Mehrsprachigkeit genutzt werden.

Die zentralen funktionalen Aspekte wie Inhalts- und Nutzerverwaltung, Zugriffssteuerung, Template Engine, ein WYSIWYG Editor und die Suchmaschine sind Teil des Kerns. Weitere Aspekte, wie die An-bindung verschiedener Authentifizierungsverfahren,[45] das Caching zur Verbesserung der Performance oder integrative Bestandteile wie XML-RPC Schnittstellen sind im Kern bereits implementiert.

FAKT: Die Installation der Core-Features hinsichtlich ihrer Funktionalität ergibt bei Joomla! eine voll nutzbare, per CMS verwaltete Website.

40 http://de.wikipedia.org/wiki/Single_Point_of_Failure
41 Components sind Apps, meist einschließlich ihrer Konfigurationsfunktionalität
42 Modules sind einfachere Komponenten zum Seitenaufbau, z.B. Banner oder Menüs
43 Plugins sind teilweise sehr umfangreiche Komponenten, die auf ein bestimmtes Ereignis warten und dann in Aktion treten.
44 Sammlungen aus Vorlagen und Formatierungsanweisungen in Form von HTML und Cascading Style Sheets
45 OpenId, Gmail, LDAP

Es ist innerhalb des Core-Feature-Sets nicht möglich, eine einzelne Joomla!-Installation für mehrere Sites parallel zu nutzen (Multi-Domain). Dazu müssen vorhandene Erweiterungen verwendet werden.

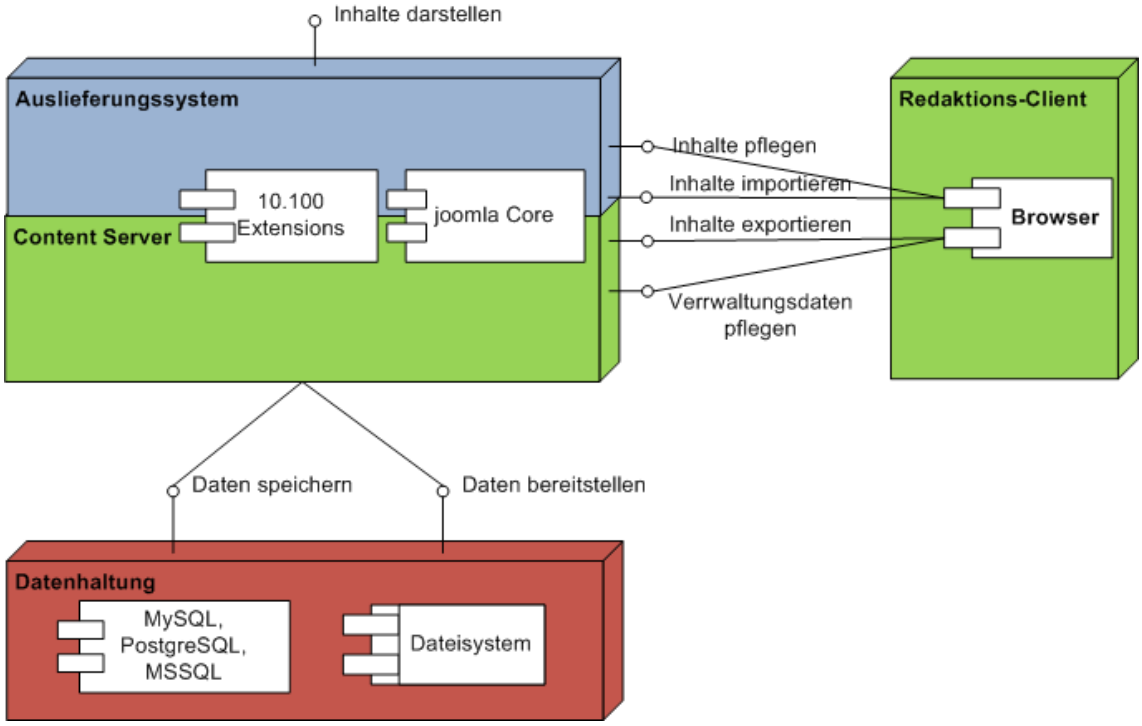


Abbildung 2.5: Joomla! – Systemarchitektur

2.2.4.3 Protokolle, Kommunikation und Datenflüsse

Die Administration, auch die Installation zusätzlicher Module erfolgt per Web-Browser über http bzw. https.

Die Kommunikation zwischen Joomla! und MySQL erfolgt über einen lokalen Socket oder über TCP (standardmäßig auf Port 3306), wie oben bei Drupal beschrieben.

2.2.4.4 Funktionalitäten

Neben den o.g. CMS Kernfunktionalitäten sind Web2.0 Bestandteile moderner Websites wie Umfragen, Newsfeeds, Nutzerbewertungen bereits im Core Feature Set enthalten. Diese Komponenten decken in der Regel den Grundbedarf einer kleinen Community-Website ab. Viele der angebotenen Erweiterungen zielen darauf ab, die bereits im Kern vorhandenen Funktionalitäten weiter zu professionalisieren oder zu vereinfachen.

So ist z.B. ein Freigabeprozess mit Versionierung der Zwischenschritte nur über Erweiterungen realisierbar. Dies deckt sich mit Drupal und WordPress. Im Joomla! Extensions Directory sind kommerzielle und Open Source Varianten dieser Erweiterung vorhanden.

Die Nutzerverwaltung ist Teil des Kernsystems und enthält bereits vordefinierte Nutzerrollen, wie zum Beispiel Autor, Editor und Publisher.

2.2.4.5 Besonderheiten

Entwickler, die eine Erweiterung oder Anpassung erstellen wollen, müssen über PHP-, SQL-, XHTML- und CSS-Kenntnisse verfügen. Skalierung und deren Optimierung sind vergleichbar zu WordPress. Im Unterschied zu WordPress ist die Erweiterung von Joomla! im Design des Systems erwünscht und die not-

wendigen APIs liegen vor, so dass mit Joomla! leicht kleinere Websites auf die Kundenwünsche zurecht geschneidert werden können.

2.2.5 TYPO3

Spezifikation	*TYPO3 CMS 6.0*
Betriebssystem	Windows, Unix, Linux
Eingesetzte Programmiersprache	PHP, Version nicht näher bezeichnet
Webserver / Applikationscontainer	Apache empfohlen mit Modulen mod_gzip/mod_rewrite, Microsoft IIS
Datenhaltung / Datenbank	MySQL, Oracle, PostgreSQL: MSSQL alles durch PHP, Versionen nicht näher bezeichnet
Redaktions-Client	Mozilla Firefox (keine Versionsangabe), IE 7+
Lizensierung	Open Source unter GPLv2 oder später ergänzt um Apache Contributor License Agreement (CLA) für Module

Tabelle 2.5: TYPO3 – Spezifikationsübersicht

2.2.5.1 Systemarchitektur

Architektonisch vertraut TYPO3 auf die Eigenschaften des LAMP Stacks. Die Performance-Optimierung des Systems besteht aus Caching Proxies, MySQL Tuning, PHP Tuning und Betriebssystem-Optimierung.

TYPO3 Instanzen können wie bereits für Drupal beschrieben, mit Betriebssystemmitteln geclustert werden. Content Server und Auslieferungsserver fallen auch hier zusammen.

2.2.5.2 Komponenten

TYPO3 ist vom Konzept her eher ein Framework. Die modulare Architektur (vgl. Abbildung 2.6) erleichtert die Anpassung und das Ersetzen einzelner Dienste, wodurch es viele optionale Plugins gibt, die wesentliche Funktionalitäten erbringen wie z.B. die Vorschau-Funktionalität.

Die zentralen funktionalen Aspekte wie Inhalts- und Nutzerverwaltung, Zugriffssteuerung, Template Engine, mehrere komfortable Rich-Text-Editoren und die Suchmaschine sind Teil des Kerns. Auch an einige sehr praktische administrative Hilfen, wie den Vergleich der Rechte zweier Benutzer oder die Beschränkung des Logins auf bestimmte IP-Adressen wurde gedacht. Man kann auch bei TYPO3 von einer bereits in Kernfunktionalität voll nutzbaren per CMS verwalteten Website sprechen.

Weitere Aspekte, wie die LDAP Anbindung oder integrative Bestandteile wie eine XML-RPC Schnittstelle sind über Erweiterungen verfügbar.

Mit Jackalope[46] ist ein Content Repository nach JSR-170/283 Standard implementiert worden. Darüber hinaus sind Interaktionen mit Java Komponenten möglich.

Die Aktualisierung bzw. Erweiterung des TYPO3 Systems erfolgt über den Extension-Manager per Web-Browser. Hier besteht ein besonderer Schutz gegen unberechtigte Manipulation der Seite mit Administrationsrechten: Die Installation setzt eine Datei ENABLE_INSTALL_TOOL voraus, die der Administrator auf dem Filesystem des Servers erzeugen muss. Die Datei wird von der Installation selbst wieder gelöscht, so dass sie vor einer neuen Erweiterung wieder erzeugt werden sollte.

46 https://fosswiki.liip.ch/display/jackalope/Home

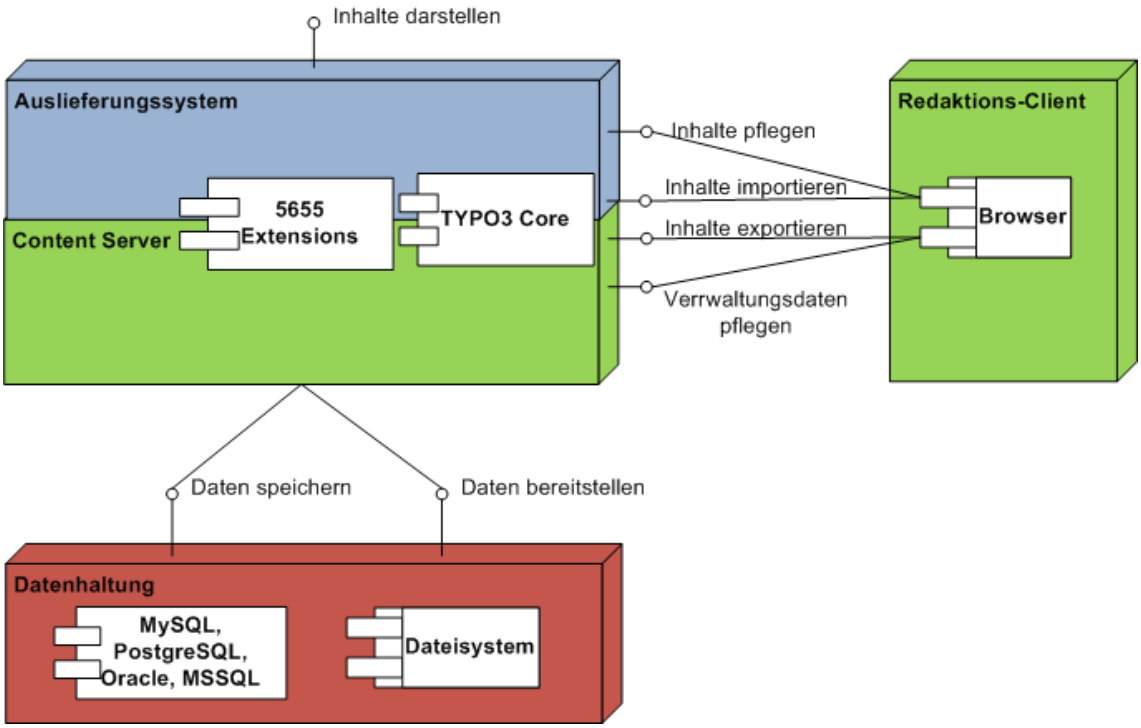


Abbildung 2.6: TYPO3 – Systemarchitektur

2.2.5.3 Protokolle, Kommunikation und Datenflüsse

Die Administration erfolgt per Web-Browser über http bzw. https. Die Kommunikation zwischen TYPO3 und MySQL erfolgt über einen lokalen Socket oder über TCP (standardmäßig auf Port 3306), wie bereits bei Drupal beschrieben.

2.2.5.4 Funktionalitäten

Die Benutzeroberfläche ist für über 40 Sprachen internationalisiert und bringt über Erweiterungen schon viele Web2.0 Elemente[47] mit sich, die in einer neuen Website nicht neu implementiert werden müssen.

FAKT: Es gibt bei TYPO3 mehrere komfortable Editoren und einen zweistufigen Redaktionsworkflow im Kernsystem.

Administratoren können mehrere Domains und mehrere Sites verwalten. Der Administrator kann für Preproduktionssysteme eine Sandbox einrichten, d.h. einen Bereich auf der Website, in dem neue Templates erprobt werden, deren Fehlfunktion die übrige Website nicht beeinflusst.

Die Zugriffsverwaltung ist gruppen- und rollenbasiert.

Die Versionierung erlaubt für den Redakteur eine unbegrenzte Undo-History.

Für einen Benutzer ist die Login-Historie zu erkennen, d.h. seine erfolgreichen und erfolglosen Anmelde-versuche einschließlich der zugehörigen IP-Adressen.

47 Foren, Chat, Kalender, Umfrage, Subscriptions, User Contributed Content, Wiki etc.

2.2.5.5 Besonderheiten

TYPO3 hat seit 2008 eine etwas unklare Release-Politik durchlebt. Die technische Struktur der Komponenten, ihre Kommunikationsbeziehungen und ihre gegenseitigen Absicherungsmöglichkeiten stehen im Vordergrund, wo möglich wird auch die interne Softwarearchitektur untersucht.

FAKT: Die traditionelle TYPO3-Linie wird jetzt als CMS 6 weiterentwickelt, während das „next generation CMS" TYPO3 Neos von Grund auf neu entwickelt wird.

Es gibt in etwa 30 Core Entwickler[48], wobei die Entwicklergemeinde vorwiegend in Deutschland beheimatet ist. In Deutschland gibt es auch sehr viele kleine Firmen, die TYPO3-Support anbieten.

Die Erzeugung von Templates erfordert einen Entwickler mit TypoScript Kenntnissen. Diese Sprache wird von PHP interpretiert, so dass im Betrieb von TYPO3 Systemen eine höhere Grundlast und höherer Speicherbedarf entsteht. Der Speicherbedarf entspricht in etwa dem von typischen JEE Anwendungen.

2.3 Aufstellung der Bewertungskriterien

2.3.1 Überblick

Die Auswahl, die Integration und der permanente Betrieb des Content Management Systems lässt sich anhand der ITIL Phasen „Service Design", „Service Transition" und „Service Operation" strukturieren. Alle drei Phasen enthalten Steuerungselemente, die für die sichere Verwendung der CMS von Bedeutung sind. Um ein Beispiel zu geben: Ein kleines Unternehmen wird ein CMS nur dann sicher verwenden können, wenn nicht nur dessen Systemarchitektur geeignet ist, die Infrastruktur gehärtet ist und die Softwarequalität des CMS herausragend, sondern wenn auch der Mechanismus zum Updaten des laufenden Systems so klar, intuitiv und fehlertolerant umgesetzt ist, dass der Administrator vorher eingerichtete Sicherheitssteuerungen nicht wieder unbeabsichtigt deaktiviert. Die sichere Benutzung ist nur dann möglich, wenn die Kette der Sicherheitselemente lückenlos über den gesamten Lebenszyklus reicht.

Unterhalb der ITIL Phasen werden im Folgenden die Qualitätskriterien des Open Software Assurance Maturity Modells (Open SAMM [49]) verwendet.

48 Eine andere Quelle zählt 15 Mitglieder des „Server Teams" + Administratoren

49 „Software Assurance Maturity Model (SAMM): A guide to building security into software development"; http://www.opensamm.org; Zugriff am 07.12.2012.

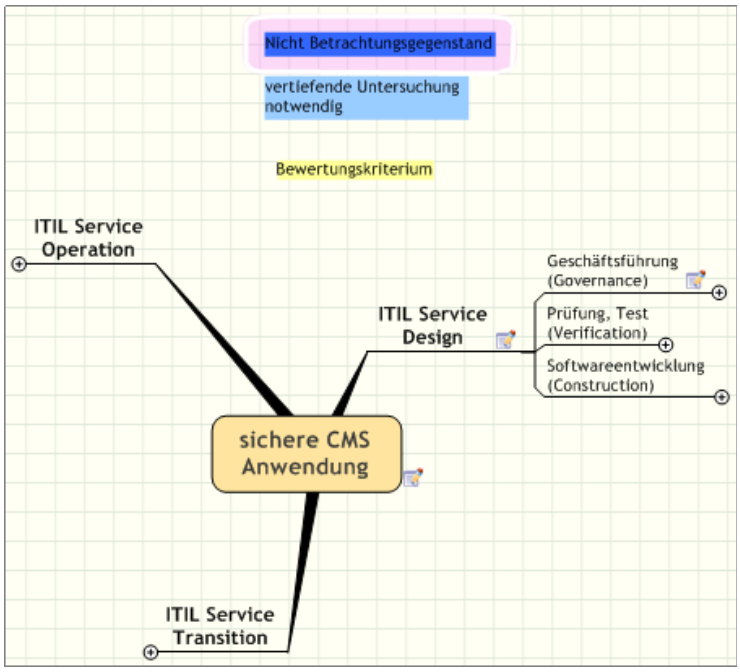


Abbildung 2.7: Kriterienüberblick

Open SAMM ist ein Ergebnis des Open WebApp Security Projektes und bedient sich bei der Formulierung der Sicherheits- und Verifikationsmechanismen der Tools und Standards (vgl. OWASP Application Security Verification Standard ASVS 2009[50]), die gleichfalls in diesem Projekt geschaffen werden.

Die Bewertungskriterien werden für alle drei ITIL Phasen in Form von Tabellen mit folgenden Spalten aufgeführt:

- Bewertungskriterien: Dies ist der Pfad aus der jeweiligen Mindmap vom Knoten „sichere CMS Anwendung“ bis zum Kriterium selbst.
- Beschreibung/Erläuterung: eine Erläuterung, was unter dem Kriterium zu verstehen ist, wenn der Name nicht selbsterklärend ist.
- Bewertung: um das Kriterium messbar zu machen, wird ein Bewertungsmaßstab gegeben. Dieser reicht in der Regel von:
  - (-) für semantisch „sicherheitstechnisch bedenklich“ über
  - (0) für „sicherheitstechnisch neutral“ bis zu
  - (+) für „sicherheitstechnisch gut oder besser“.

Einige Kriterien sind so dargestellt, dass ihre Erfüllung aus Sicht der IT-Sicherheit maximal den Normalzustand repräsentiert. Dies können auch Ausschlusskriterien sein, die in dieser Spalte dann bei Erfüllung nur eine (0) erhalten. Den anderen Fall gibt es natürlich auch: Kriterien, die eine positive Abweichung vom Allgemeinzustand darstellen, erhalten bei Nichterfüllung eine (0) und bei Erfüllung ein (+).

50 https://www.owasp.org/index.php/Projects/OWASP_Application_Security_Verification_Standard_Project/Releases/ASVS_-_2009_Edition

- Ausschlusskriterium: Alle Kriterien, die so wichtig sind, dass ein System aus sicherheitstechnischer Sicht den Mindestwert erreichen muss, sind als Ausschlusskriterium markiert.
- Wichtung: Die Spalte Wichtung repräsentiert die Bedeutung des Kriteriums für die Produktauswahl. Die Einschätzung der Wichtung muss individuell überdacht werden. Beispiel: für eine kleine Kommune mit ausschließlich deutsch sprechendem Administrator ist das Kriterium „Mehrsprachigkeit“ (der Dokumentation) sehr wichtig, während es hier mit „gering“ bewertet wurde. Die Wichtung ist folglich als Maß dafür zu verstehen, wie viel Einfluss das Kriterium auf die sichere CMS Anwendung hat. Selbstverständlich kann es Kriterien geben, deren Wichtung mit „hoch“ angegeben ist, deren Nichterfüllung jedoch nicht zum Ausschluss des CMS führen müssen.

2.3.2 Service Design

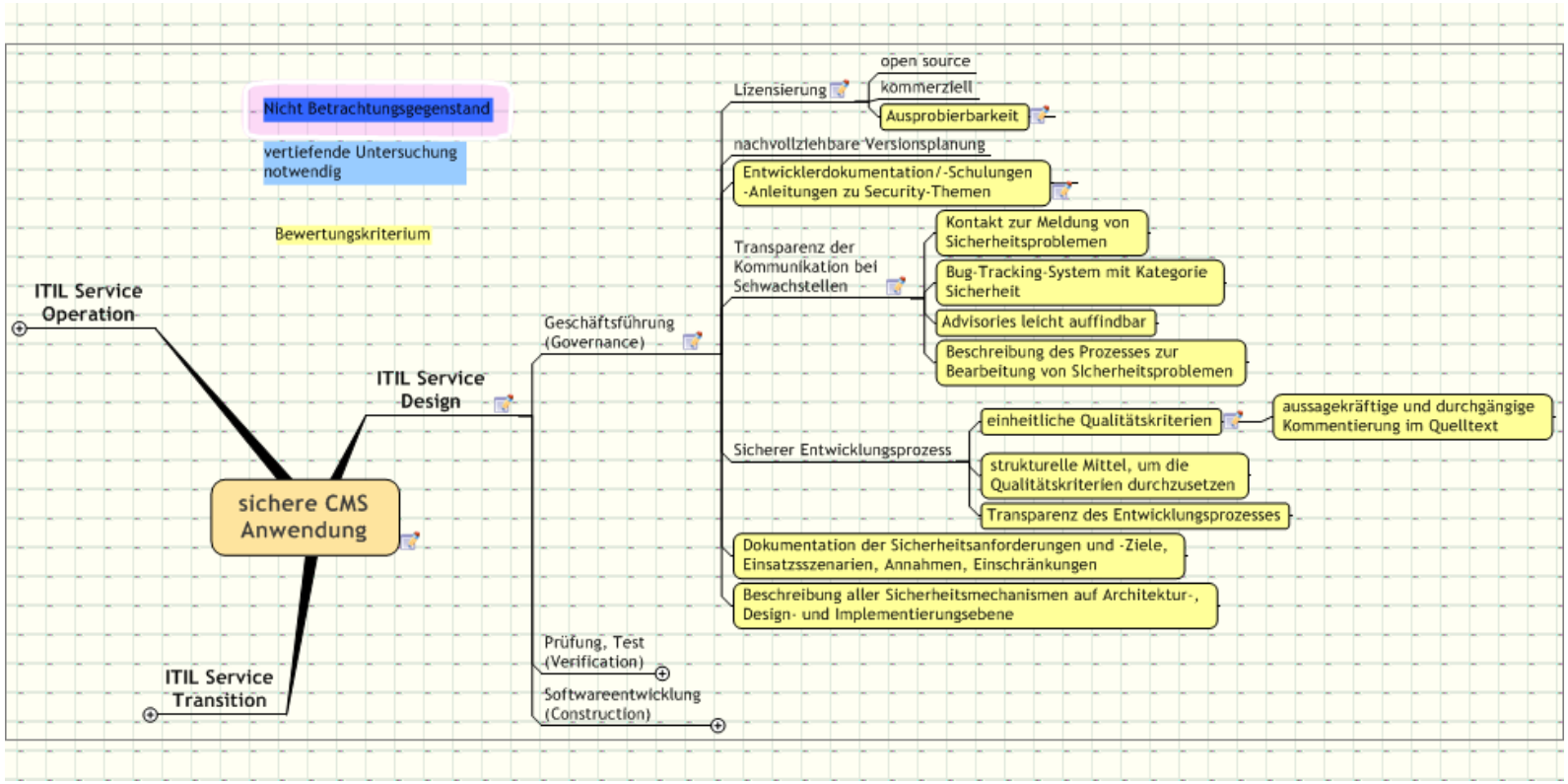


Abbildung 2.8: Kriterienüberblick – Service Design (Geschäftsführung)

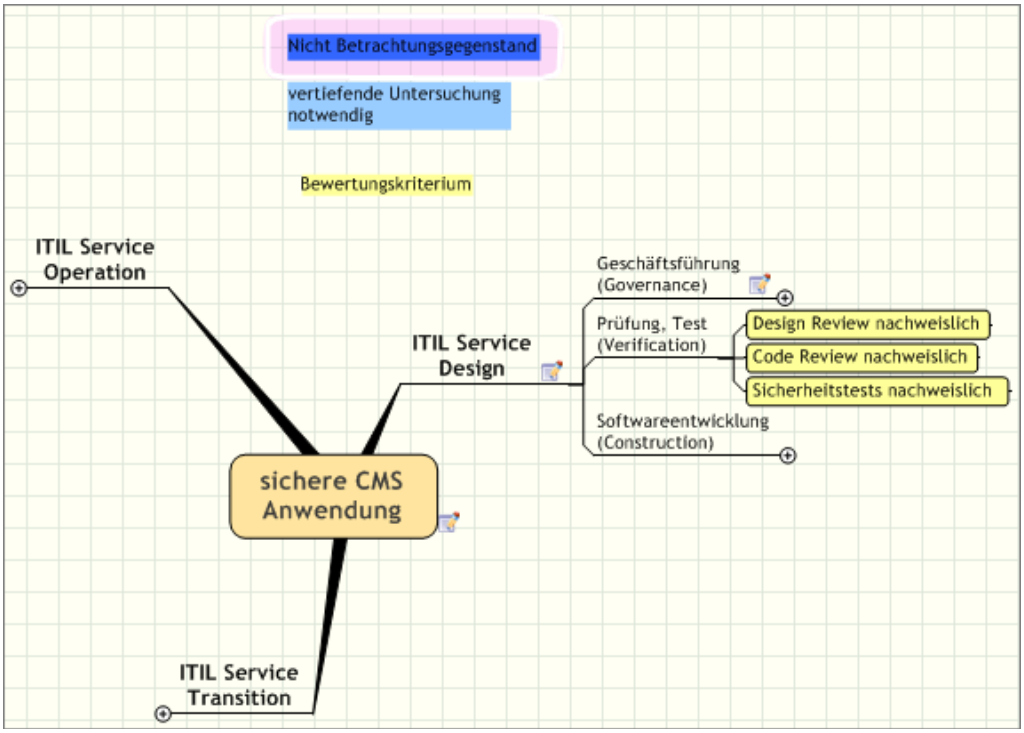


Abbildung 2.9: Kriterienüberblick – Service Design (Prüfung, Test)

Wie aus Abbildung 2.9 ersichtlich, sind für die Verifikation der Umsetzung der Sicherheitsanforderungen zahlreiche Tests (automatisierte und manuelle Quellcodereviews, Schwachstellentests) notwendig, die jeweils individuell durchgeführt werden müssen.

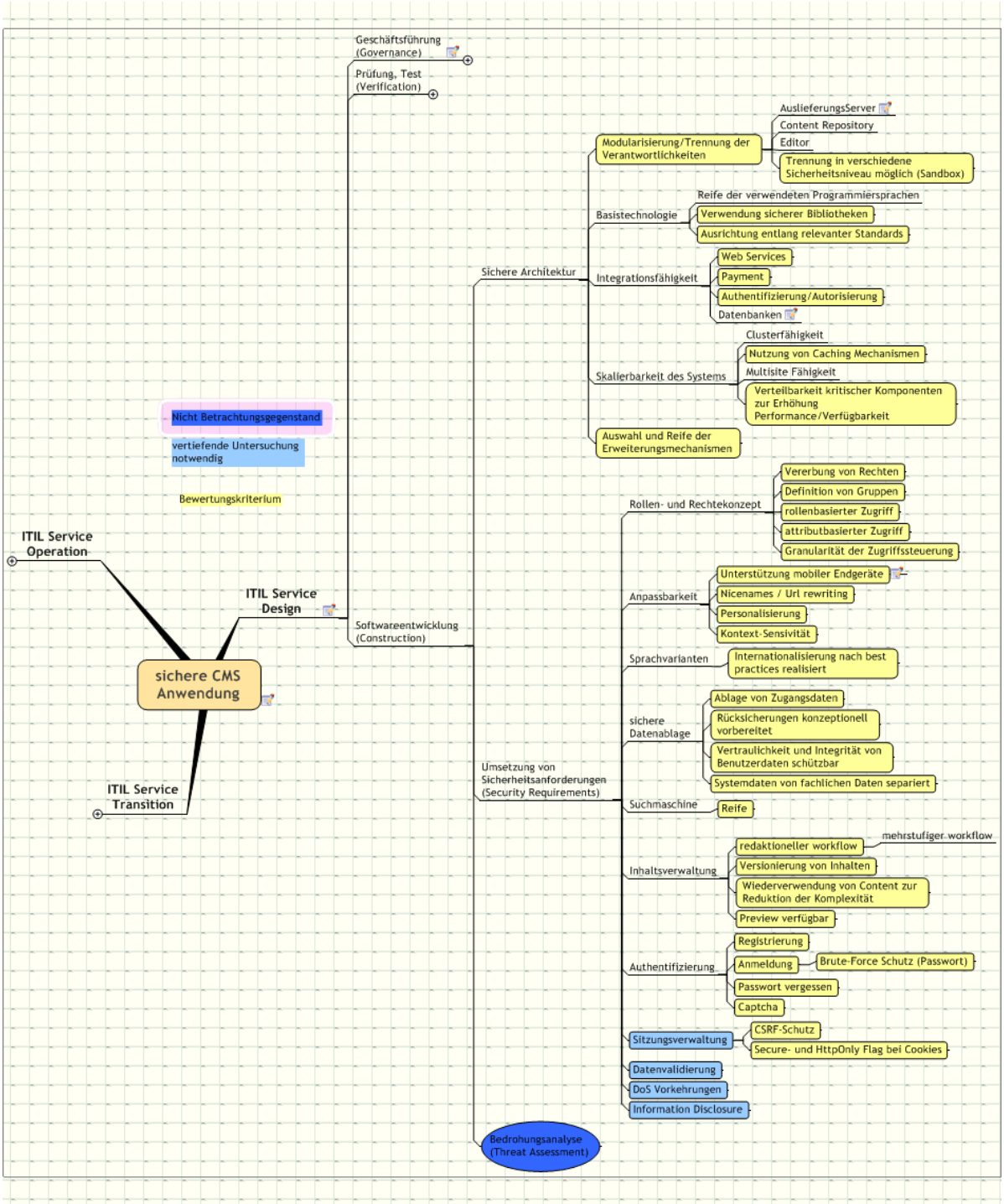


Abbildung 2.10: Kriterienüberblick – Service Design (Softwareentwicklung)

Service Design						
Bewertungskriterien			***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
Geschäfts-führung (Governance)	Lizenzierung	Ausprobierbarkeit	Potentielle Dienstanbieter können leicht eine Instanz des CMS mit Beispieldaten aufsetzen, um die Installation, den Update-Mechanismus, die Anforderungen des Systems an die Administrationskenntnisse, die Angemessenheit der Lösung für die konkreten Geschäftsanforderungen zu prüfen. Leichte Ausprobierbarkeit wirkt darauf hin, dass das System weithin in Benutzung = im (Sicherheits-)Test ist.	(-) Anmeldung notwendig, um einen Link zu bekommen. (+) Es besteht ein Download Link.	ja	mittel
	Entwicklerdokumentation/-Schulungen/ Anleitungen zu Security-Themen		Es gibt konkrete Hilfestellungen zum Thema Sicherheit für Entwickler des CMS und für Dienstanbieter (Entwickler, Administratoren), die das CMS, dessen APIs oder Komponenten nutzen.	(-) Es gibt Verweise auf relevante Dokumente. (0) Es gibt konkrete Dokumente zu Security-Themen. (+) Es gibt Code- und/oder Konfigurationsbeispiele.	ja	hoch
	Transparenz der Kommunikation bei Schwachstellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	Die Bekanntgabe einer schnell auffindbaren Kontaktadresse ist Minimum.	(0) ja	ja	hoch
		Bug-Tracking-System mit Kategorie Sicherheit	Die Sichtbarkeit der aufgetretenen Probleme, deren Bearbeitungszustand und	(0) ja	ja	hoch

Service Design						
Bewertungskriterien			***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			-tempo kann ausschlaggebend für das Vertrauen in ein Produkt sein.			
		Advisories auf der Website des Herstellers leicht auffindbar	Die Sichtbarkeit der aufgetretenen Probleme, deren Bearbeitungszustand und -tempo kann ausschlaggebend für das Vertrauen in ein Produkt sein.	(0) ja	ja	hoch
		Beschreibung des Prozesses zur Bearbeitung von Sicherheitsproblemen	Sicherheitsprobleme sind überall an der Tagesordnung. Entscheidend ist, wie sich ein Dienstanbieter darauf vorbereiten kann, dass bei ihm selbst Handlungsbedarf entsteht. Transparenz der Handlungen im Notfall schaffen Sicherheit.	(0) ja	ja	hoch
	Sicherer Entwicklungsprozess	einheitliche Qualitätskriterien	Transparente Kriterien sind zumindest notwendige Voraussetzung für einheitliche (Mindest-)Qualität.	Existieren nachweislich verbindliche Kriterien, nach denen die Entwickler die Software erstellen? (+) ja (-) nein		hoch
		aussagekräftige und durchgängige Kommentierung im Quelltext	Kommentierung hilft anderen Entwicklern, die den Code weiterentwickeln oder nutzen wollen. Damit kann sich das Review auf die wirklich wichtigen Aspekte	(+) ja (-) nein		mittel

Service Design						
Bewertungskriterien			***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			konzentrieren. Bei hinreichender Kommentierung werden Fehler vermieden, die dadurch entstehen, dass der Kontext nicht richtig verstanden wurde.			
		strukturelle Mittel, um die Qualitätskriterien durchzusetzen	Das Vorhandensein gemeinsamer Regeln reicht oft nicht. Bestehen tatsächliche Mittel organisatorischer oder technischer Art, die Einhaltung der Regeln umzusetzen und wird dies auch getan?	(+) ja (-) nein		hoch
		Transparenz des Entwicklungsprozesses	Wenn für Experten ersichtlich ist, wie aufgetretene Schwachstellen bearbeitet wurden, kann dies öffentlich geprüft werden. Davon profitieren auch die Dienstanbieter, die allein nie in der Lage wären, eine solche Prüfung durchzuführen. Transparenz ermöglicht dem Dienstanbieter[51], unabhängig zu bewerten, welche Sorgfalt dem Thema Sicherheit beigemessen wird, welche Entwickler beteiligt sind, wie lange das Review gedauert hat etc.	(-) Es ist nicht ersichtlich, wie entwickelt wird und wie ein Bug behoben wurde. (0) Es ist ersichtlich, was geändert wird. (+) Es ist ersichtlich, welche Bugs es gibt und wie die Bearbeitung der Bugs fortschreitet.		hoch

51 vgl. Definition vom „Dienstleister“ im Stichwortverzeichnis

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
	Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen			Die klare Spezifikation der Anforderungen, für die das CMS konzipiert wurde lässt sich mit den Anforderungen des Dienstanbieters vergleichen.	(+) ja (-) nein		gering
	Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene			Die klare Spezifikation der Lösung ermöglicht das externe Review.	(+) ja (-) nein		hoch
Prüfung/Test (Verification)	Design Review nachweislich			Lässt sich ein Nachweis finden, dass entsprechende Reviews regelmäßig oder überhaupt stattfinden?	(+) ja (-) nein		hoch
	Code Review nachweislich			Lässt sich ein Nachweis finden, dass entsprechende Reviews regelmäßig oder überhaupt stattfinden?	(+) ja (-) nein		hoch
	Sicherheitstests nachweislich			Lässt sich ein Nachweis finden, dass Sicherheitstests regelmäßig oder überhaupt stattfinden?	(+) ja (-) nein		hoch
Softwareentwicklung (Construction)	Sichere Architektur	Modularisierung / Trennung der logischen Funktionen	Modularisierung / Trennung der logischen Funktionen	Die Trennung der Verantwortlichkeiten ist ein Grundprinzip verteilter Softwareentwicklung. Die voneinander getrennten Bestandteile weisen geringere Komplexität auf und lassen sich leichter noch zielgerichteter absichern.	(0) Trennung auf Serverebene (+) Trennung auf Komponentenebene		mittel
			Trennung in ver-	Die Trennung der vertrauens-	(+) Built-in	ja	hoch

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			schiedene Sicherheitsniveaus möglich (Sandbox)	würdigen von den nicht vertrauenswürdigen Systembestandteilen ist ebenfalls ein allgemeiner Ansatz. Hierfür gibt es verschiedene Mittel, z.B. Java Sandbox, Virtualisierung auf Betriebssystemebene, Rechtebeschränkung auf Betriebssystemebene, Sandboxing im CMS selbst.	(0) über Betriebssystemmittel		
		Basistechnologie	Verwendung sicherer Bibliotheken	Prüfung der Installation auf veraltete Softwareversionen	(0) Es werden keine veralteten, unsicheren Bibliotheken in der Standard-Installation gefunden	ja	hoch
			Ausrichtung entlang relevanter Standards	Werden Standards wie RSS, XHTML, jquery, WebDAV, WS-Security, CMIS unterstützt, oder werden eigene freie Lösungen angeboten?	(+) Standards werden verwendet (-) Prioprietäre Lösungen werden angeboten		mittel
		Integrationsfähigkeit	Web Services	Wird WS-Security unterstützt ? Werden SOAP Web Services unterstützt?	(+) Unterstützung WS-I Basic Security Profile, Anbindung über Standardbibliotheken (0) Anbindung über eigene Bibliotheken (-) Keine Anbindung möglich		mittel
			Payment	Sind die Anbindungen an das Payment-Verfahren an bestehenden Standards ausgerichtet, z.B. SET oder muss schlimmstenfalls der Dienst-	(+) Anbindung über Standardbibliotheken (0) Anbindung über eigene Bibliotheken (-) Keine Anbindung möglich		mittel

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
				anbieter eigenes entwickeln.			
			Authentifizierung/ Autorisierung	Unterstützung der Verfahren OpenId, Oauth, LDAP im Kernsystem oder durch stabile Erweiterungen	(+) alle drei (0) zwei (-) eins	ja	mittel
		Skalierbarkeit des Systems	Nutzung von Caching Mechanismen		(+) architektonisch oder CMS spezifisches Caching (0) Standard-Mittel		mittel
			Verteilbarkeit kritischer Komponenten zur Erhöhung Performance/ Verfügbarkeit		(+) vertikal und/oder horizontal verteilbar (-) keine Verteilbarkeit		mittel
		Auswahl und Reife der Erweiterungsmechanismen		Sind Erweiterungen architektonisch vorgesehen und wie sind sie gelöst?	(+) Saubere Trennung der APIs nach Erweiterungsaspekten (0) Erweiterungen über definierte APIs (-) keine Erweiterungsmöglichkeiten		mittel
	Umsetzung von Sicherheitsanforderungen (Security Requirements)	Rollen- und Rechtekonzept	Vererbung von Rechten	Das Content-Modell sieht vor, dass Elemente die Rechte übergeordneter Elemente erben können, so dass nicht jedes Element explizit mit Rechten belegt werden muss.	(+) ja (-) nein		mittel
			Definition von Gruppen	Zuordnung von Benutzern zu Benutzergruppen, über die gemeinsam eine Zugriffssteuerung definiert werden	(+) ja (-) nein		mittel

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
				kann			
			rollenbasierter Zugriff	Zuordnung von Benutzern zu Rollen, über die gemeinsam eine Zugriffssteuerung definiert werden kann	(+) ja (-) nein		mittel
			attributbasierter Zugriff	Die Autorisierungsentscheidung zum Zugriff auf eine Ressource kann von einzelnen Eigenschaften des Nutzers und der Ressource abhängig sein, Bsp: Rechnungen mit Auftragswert>100.000 Euro darf Nutzer Alice nicht sehen.	(+) ja (-) nein		gering
			Granularität der Zugriffssteuerung	Lassen sich Inhaltsbereiche einer Webseite hinsichtlich unterschiedlicher Zugriffsrechte gliedern?	(+) ja (-) nein		mittel
		Anpassbarkeit	Unterstützung mobiler Endgeräte	Ist responsive Design mit Bordmitteln oder mit Erweiterungen umsetzbar?	(+) ja (-) nein		mittel
			Nicenames / URL Rewriting	Gibt es die Möglichkeit, Nicenames zu nutzen? Nicenames erhöhen die Wartbarkeit	(+) ja (-) nein		gering
			Personalisierung	Ist Personalisierung umsetzbar?	(+) ja (-) nein		gering
			Kontext-Sensitivität	Ist der Inhalt abhängig vom Kontext des Benutzers, der Situation anzuzeigen/auszu-	(+) ja (-) nein		gering

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
				blenden?			
		Sprachvarianten	Inter-nationalisierung nach Best Practices realisiert	Wird UTF-8 verwendet?	(+) ja (-) nein		mittel
		sichere Daten-ablage	Ablage von Zu-gangsdaten	Gibt es hier Verfahren nach Best Practices ? Sind mindestens die von der Bundesnetzagentur empfohlenen Algorithmen und Parameter nutzbar?	(+) ja (-) nein		hoch
			Rücksicherungen konzeptionell vor-bereitet	Ist die Rücksicherung einzelner Bereiche des Datenbestandes möglich?	(+) ja (-) nein		mittel
			Vertraulichkeit und Integrität von Be-nutzerdaten bzw. Vorgangsdaten schützbar	Ist die Ablage und Archivierung besonders vertraulicher Daten separat für ein höheres Schutzniveau zu konfigurieren ?	(+) ja (-) nein		mittel
			Systemdaten von fachlichen Daten separiert	Sind die Daten zum Betrieb des Systems von den fachlich relevanten Daten trennbar, so dass der technische Betrieb keine fachlichen Daten sehen muss?	(+) ja (-) nein		mittel
		Suchmaschine	Reife der Such-maschine	Wird eine Standard-Such-maschine verwendet?	(+) ja (-) nein		
		Inhaltsverwaltung	redaktioneller Workflow	Ist ein mehrstufiger Freigabe-prozess definierbar?	(+) ja (-) nein		mittel

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			Versionierung von Inhalten	Sind die Versionen eines Artikels und die getätigten Änderungen im Nachhinein feststellbar?	(+) ja (-) nein		mittel
			Wiederverwendung von Content zur Reduktion der Komplexität	Kann Content ohne Duplizierung an verschiedenen Stellen in verschiedenen Kontexten der Website verwendet werden?	(+) ja (-) nein		gering
			Preview verfügbar	Gibt es eine korrekte Preview?	(+) ja (-) nein		mittel
		Authentifizierung	Registrierung	Wie sicher ist der Registrierungsprozess?	(0) Keine konzeptionellen Schwachstellen, z.B. User-Enumeration, sichtbar	ja	hoch
			Anmeldung	Wie sicher ist der Anmeldungsprozess?	(0) Keine konzeptionellen Schwachstellen, z.B. User-Enumeration, sichtbar	ja	hoch
			Brute-Force-Schutz (Passwort)	Gibt es einen Brute-Force-Schutz für Passwörter?	(0) ja	ja	hoch
			Passwort vergessen	Wie sicher ist die Behandlung vergessener Passwörter?	(0) Keine konzeptionellen Schwachstellen sichtbar	ja	hoch
			Captcha	Sind Captchas einbindbar?	(+) verschiedene einbindbar (0) mindestens ein Verfahren verfügbar (-) nein		mittel
		Sitzungsverwaltung	CSRF-Schutz	Gibt es standardmäßig einen CSRF-Schutz?	(+) ja (-) nein		hoch
			Secure- und	Sind die Einstellungen	(0) ja	ja	hoch

Service Design							
Bewertungskriterien				***Beschreibung /Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			HttpOnly Flag bei Cookies	konfigurierbar?			

Tabelle 2.6: Kriterien – Service Design (Details)

2.3.3 Service Transition

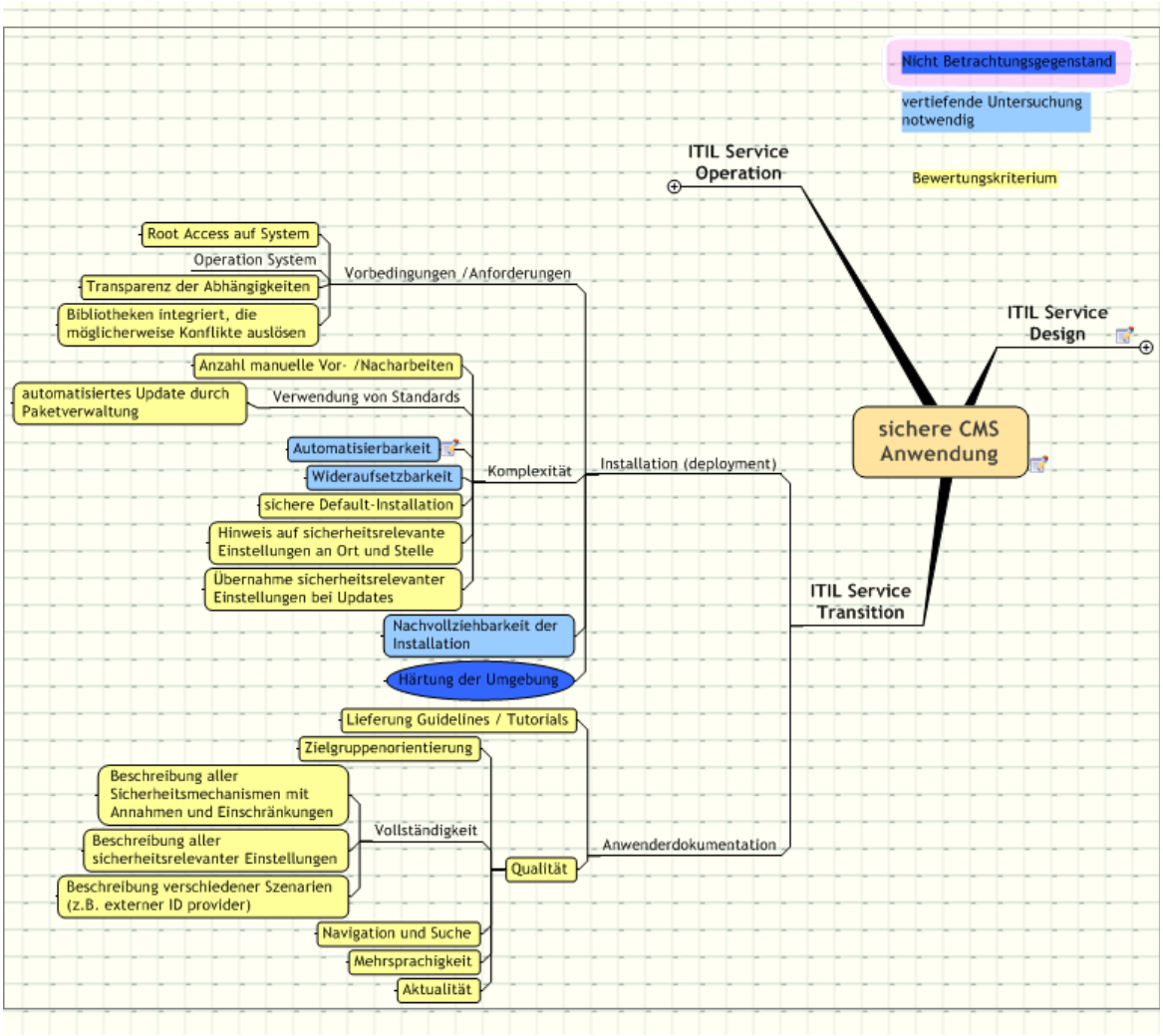


Abbildung 2.11: Kriterienüberblick – Service Transition

Service Transition							
Bewertungskriterien				***Beschreibung/Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
Installation (Deployment)	Vor-bedingungen/ Anforderungen	Root Access auf System		Sind Root Rechte erforderlich?	(-) ja (+) nein		mittel
		Transparenz der Abhängigkeiten		Gibt es Hilfsmittel (build-System wie maven oder Paketmanager oder CMS Komponente) die ersichtlich machen, welche Pakete die Installation tatsächlich nutzt?	(+) ja (-) nein		mittel
		Bibliotheken integriert, die möglicherweise Konflikte auslösen		Werden eigene Bibliotheken mitgeliefert, obwohl diese in der Umgebung bereits vorhanden sein müssten?	(-) ja (+) nein		mittel
	Komplexität	Anzahl manuelle Vor-/Nacharbeiten bei Konfiguration und Installation		Wie viele sicherheitsrelevante Schritte müssen zur Konfiguration des Systems bei Erstinstallation durchlaufen werden?	(+) unter 5 Schritte (0) 5-10 Schritte (-) 11-15 Schritte	ja	hoch
		Verwendung von Standards	automatisiertes Update durch Paketverwaltung	Paketverwaltungen wie dpkg, msi oder rpm ermöglichen eine fehlerver-meidende Installation und De-installation, in dem sie spezifizierte Ab-hängigkeiten prüfen und definierte Installationsabläufe umsetzen.	(+) ja (-) nein		mittel
		sichere Default-Installation		Werden bereits sichere Standardwerte gesetzt?	(+) ja (-) nein		mittel
		Hinweis auf sicherheitsrelevante Ein-stellungen an Ort und Stelle		Gibt es verständliche Hinweise in den Konfigurationsdateien selbst?	(+) ja (-) nein		mittel
		Übernahme sicherheitsrelevanter Ein-stellungen bei Updates			(+) ja (-) nein		hoch
Anwender-dokumentation	Lieferung Guidelines / Tutorials				(+) ja (-) nein		gering
	Qualität	Zielgruppenorientierung		Gibt es für die Zielgruppen Ad-ministratoren, Entwickler und Anwender	(+) ja (-) nein		gering

Service Transition							
Bewertungskriterien				***Beschreibung/Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
				separate Dokumentationen?			
		Navigation und Suche		Sind Struktur und Formulierung geeignet, die Informationen zu finden? Kann man punktuell darauf zugreifen? Enthält jedes Thema sinnvolle Informationen (oder gibt es Seiten, die nur weiter verweisen)?	(+) ja (-) nein		gering
		Mehrsprachigkeit		Ist sicherheitsrelevante Dokumentation auch in deutsch vorhanden?	(+) ja (-) nein		gering
		Aktualität		Ist sicherheitsrelevante Dokumentation auf dem letzten Stand?	(+) Alle Dokumentation ist bis auf vereinzelte Ausnahmen aktuell (-) Es gibt öfter veraltete Informationen	ja	hoch
		Vollständigkeit	Beschreibung aller Sicherheitsmechanismen mit Annahmen und Einschränkungen		(+) ja (0) nein		mittel
			Beschreibung aller sicherheitsrelevanten Einstellungen		(+) ja (0) nein		hoch
			Beschreibung verschiedener Szenarien (z.B. externer ID Provider)		(+) ja (0) nein		gering

Tabelle 2.7: Kriterien – Service Transition (Details)

2.3.4 Service Operation

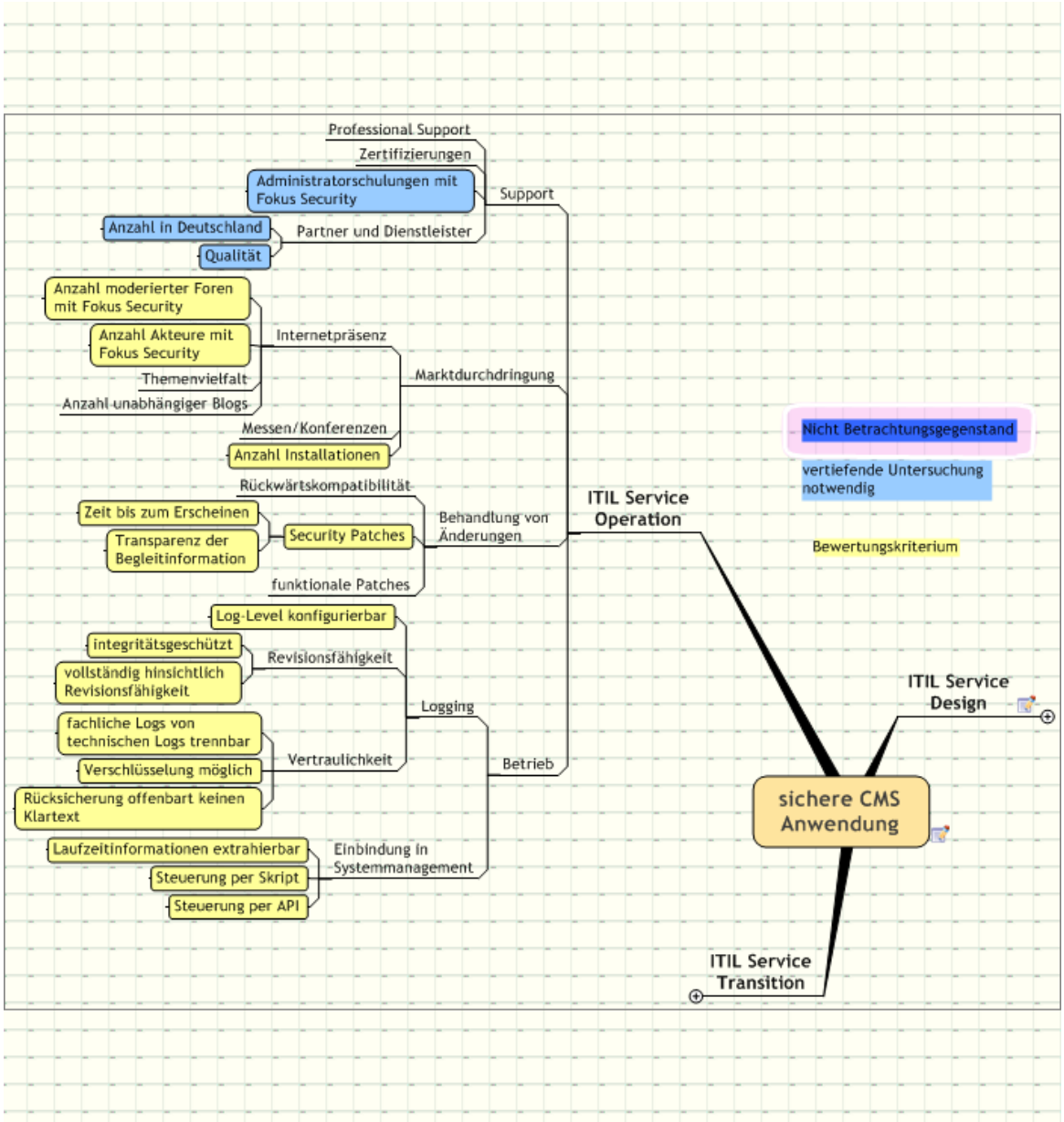


Abbildung 2.12: Kriterienüberblick – Service Operation

Service Operation							
Bewertungskriterien				***Beschreibung/Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
Markt-durchdringung	Internet-präsenz	Anzahl moderierter Foren mit Fokus Security		Die Foren sind sehr praxisnah und damit eine gute Ergänzung zur Dokumentation	(+) mindestens eins (-) kein Forum		hoch
		Anzahl Akteure mit Fokus Security		Anzahl der Mitarbeiter, bzw. der Entwickler im Security Team, die tatsächlich die Bugs beheben und Schwachstellen suchen.	(+) mehr als 10 (0) zwischen 5 und 10 (-) weniger als 5	ja	hoch
	Anzahl Installationen			in Deutschland	(+) mehr als 50 (0) zwischen 10 und 50 (-) weniger als 5	ja	mittel
Behandlung von Änderungen	Security Patches	Zeit bis zum Erscheinen		Nur Schwachstellen sehr hoher Priorität, um vergleichbar zu sein.	(+) weniger als 2 Tage (0) zwischen 2 und 8 Tagen (-) mehr als 8 Tage	ja	hoch
		Transparenz der Begleitinformation			(+) Es ist übersichtlich gegliedert, es ist eine nachvollziehbare Problembeschreibung vorhanden (0) Es ist erkennbar, wer betroffen ist. (-) andernfalls		hoch
Betrieb	Logging	Revisions-fähigkeit	integritätsgeschützt	Können die Logs mit Zeitstempel und Signatur versehen werden?	(+) ja (-) nein		mittel
			vollständig hinsichtlich Revisionsfähigkeit	Sind fachliche Logs konfigurierbar, aus denen hervorgeht, wer wann welche Aktion ausgelöst hat?	(+) ja (-) nein		mittel
		LogLevel konfigurierbar		Sind verschiedene Loglevel zwischen Trace und Info konfigurierbar	(+) ja (-) nein		mittel
		Vertraulich-keit	fachliche Logs von technischen Logs	Kann das Logging so konfiguriert werden, dass die technischen Administratoren keine	(+) ja (-) nein		mittel

Service Operation							
Bewertungskriterien				***Beschreibung/Erläuterung***	***Bewertung***	***Ausschluss-kriterium***	***Wichtung***
			trennbar	vertraulichen fachlichen Informationen sehen?			
			Verschlüsselung der fachlichen Logs mit Bordmitteln möglich	Kann das Logging so konfiguriert werden, dass die technischen Administratoren keine vertraulichen fachlichen Informationen sehen?	(+) ja (-) nein		mittel
			Rücksicherung offenbart keinen Klartext	Könnte die Rücksicherung von Informationen aus der Archivierung so gestaltet werden, dass keine vertraulichen Informationen offenbart werden?	(+) ja (-) nein		mittel
	Einbindung in System-management	Laufzeitinformationen extrahierbar		Sind die Zustände der Komponenten (Auslastung, Ressourcenbenutzung) von außen abfragbar, so dass diese Information in ein Systemmanagement mit einfließen kann?	(+) ja (-) nein		mittel
		Steuerung per Skript		Können wesentliche Funktionen des CMS (Start, Stop, Neuladen der Konfiguration) per Skript gesteuert werden?	(+) ja (-) nein		mittel
		Steuerung per API		Können wesentliche Funktionen des CMS (Start, Stop, Neuladen der Konfiguration) per API gesteuert werden?	(+) ja (-) nein		gering

Tabelle 2.8: Kriterien – Service Operation (Details)

2.4 Typische Anwendungsszenarien

Alle oben beschriebenen CMS können in verschiedenen Ausbaustufen betrieben werden. Wie komplex das CMS ist, hängt nicht zuletzt davon ab, welche Module zusätzlich installiert werden. Diese individuelle Konfiguration des CMS hat Einfluss auf die Handhabbarkeit und ist zudem aus sicherheitstechnischer Sicht relevant.

Um die Betrachtungen in den Kapiteln 4 und 5 auf eine einheitliche Grundlage zu stellen, werden hier typische Anwendungsszenarien durchgespielt. In diesen Szenarien finden sich Privatpersonen, öffentliche Stellen oder mittelständische Unternehmen wieder, welche typische Anwender der ausgewählten CMS sein könnten. Auf die Beurteilung sehr komplexer Szenarien wurde verzichtet. Dafür ist in jedem Fall eine individuelle Analyse empfehlenswert.

Auf Basis der folgenden Szenarien können die dafür notwendigen Funktionalitäten bzw. Module pro CMS definiert werden. Hiermit wird die Vergleichbarkeit in Bezug auf den technischen Umfang sichergestellt, ohne dass typische Anwendungsfälle übergangen werden.

2.4.1 Szenario 1: „Private Event Site“

Das Szenario entspricht der oft genutzten Möglichkeit, für einen Anlass im privaten Umfeld – Geburtstag, Party, Hochzeit o.ä. – eine eigene, meist temporäre Website zu nutzen. Die Zielgruppe sind Internet affine Freunde und Bekannte des Ausrichtenden. Die Website soll „schick“ aussehen und im übrigen wenig Arbeit machen.

Redakteure, die für die Erstellung der Inhalte verantwortlich sind, kommen aus dem Freundeskreis des Ausrichtenden. Die Erstellung der Inhalte erfolgt ad hoc ohne definierte Redaktionsworkflows. Typische Inhalte sind eine Anfahrtsskizze, einige private Fotos sowie Texte, die das Event und einige Details beschreiben. Ein Wetterservice des Veranstaltungsortes soll eingebunden sein. Die Gäste sollen sich im Gästebuch eintragen können. Anschließend soll es einzelnen Gästen möglich sein, private Fotos hochzuladen.

Da sich niemand die Mühe machen möchte, Benutzerkennungen für die Gäste zu vergeben, gibt es Kommentare im Gästebuch nur von anonymen Benutzern. Der Ausrichter möchte trotzdem verhindern, dass automatisierte Internet-Bots dort unpassende Eintragungen vornehmen.

Der Dienstanbieter verfügt gerade über so viel technische Erfahrung, um eine Installation der CMS Komponenten durchzuführen. Für ihn ist wichtig, dass die Website zu einem bestimmten Termin funktionsfähig und während des Events verfügbar ist. Zudem soll sie anschließend für einige Wochen online sein.

Die Architektur des Gesamtsystems ist für den Dienstanbieter unwichtig. Die Site ist für weniger als zehn parallele Nutzer konzipiert.

2.4.2 Szenario 2: „Bürgerbüro einer kleinen Gemeinde“

Das Szenario (2) beschreibt Websites für kleine kommunale oder mittelständische Dienstanbieter mit geringen finanziellen Ressourcen zur redaktionellen und technischen Umsetzung der geforderten Website.

Die Pflege und der Betrieb der CMS Komponenten liegt in Personalunion bei einem Mitarbeiter. Für die Pflege der Inhalte steht eine halbe Stelle zur Verfügung.

Die Website umfasst dabei ca. 50–100 Seiten. Die Bearbeitung erfolgt über einen Editor, der Fokus liegt hier mehr auf der Änderung bestehender Seiten und dem gelegentlichen Hinzufügen neuer Dokumente.

Die Website enthält einen geschützten Bereich für eine berechtigte Nutzergruppe. Innerhalb des Bereichs werden personenbezogene Daten dargestellt.

Die Website erlaubt Basis-Interaktionen wie beispielsweise das Ausfüllen eines Kontaktformulars und eine Suche.

Ein Fremdsystem, wie zum Beispiel die Nutzerverwaltung der Mitarbeiter des Dienstanbieters ist an das CMS System angekoppelt.

Ein konkretes Beispiel für dieses Szenario wäre ein kommunales Bürgerbüro, bei der sich das Angebot sowie die Öffnungszeiten ansehen und Termine mit Sachbearbeitern vereinbaren lassen.

2.4.3 Szenario 3: „Open Government Site einer Kleinstadt“

Das Szenario (3) beschreibt Websites mit zusätzlich zu den im Szenario (2) beschriebenen Funktionalitäten wie der Anbindung von Geschäftsprozessen und der Einbindung einer Community. Im Szenario (3) ist die Anbindung von ca. 2-3 Fremdsystemen gefordert.

Zur technischen Pflege der CMS-Komponenten stehen finanzielle Mittel für zwei bis drei Mitarbeiter mit IT-Wissen zur Verfügung und mindestens eine Stelle für Öffentlichkeitsarbeit. Die Anzahl der zu pflegenden Seiten liegt bei über 100 Seiten. Die Website verfügt über Mechanismen zur Bürgerbeteiligung wie Umfragen, Foren, Newsletter, User Generated Content. Sie dient als zentrale Plattform, über die Informationen ausgetauscht und Transaktionen aus Sicht des Bürgers abgewickelt werden.

Die Website wird mehrsprachig angeboten. Aufgrund der Menge der zu verwaltenden Inhalte und Sprachvarianten sind das Verwenden von Vorlagen (Templates) und Dokumententypen zwingender Bestandteil des CMS.

Die Anbindung der Geschäftsprozesse erfolgt über standardisierte Schnittstellen (Web-Services) der Fachverfahren.

Ein konkretes Beispiel für diesen Fall wäre eine mittlere Kommune, bei der die Bürger über aktuelle Vorhaben der Öffentlichkeit diskutieren können, Anträge an ihre Verwaltung stellen und sich über deren aktuellen Status informieren können.

2.4.4 Szenario 4: „Mittelständisches Unternehmen mit mehreren Standorten“

Das Szenario orientiert sich an einem Dienstanbieter, der mit einem eingesetzten CMS mehrere Websites für unterschiedliche Zielgruppen mit gemeinsam genutzten Inhalten umsetzen möchte.

Redakteure, die für die Erstellung der Inhalte verantwortlich sind, kommen in diesem Fall aus unterschiedlichen Bereichen. Die Erstellung der Inhalte erfolgt z.T. nach dem Mehr-Augen-Prinzip und erfordert definierte Redaktionsworkflows.

Der Dienstanbieter verfügt über vier bis fünf Mitarbeiter für den Betrieb der CMS Komponenten sowie mindestens vier Mitarbeiter, die über Marketing/Öffentlichkeitsarbeit an den Redaktionsworkflows beteiligt sind. Die inhaltliche und technische Umsetzung ist völlig separiert. Geldmittel und Fachexpertise sind nicht nur für den Aufbau, sondern auch für die Weiterentwicklung fest für mehrere Jahre eingeplant.

Die Architektur des Gesamtsystems wird für punktuell große Benutzerzahlen skalierbar konzipiert, um terminbasierte Marketingaktionen durchzuführen.

Mehr als drei Fremdsysteme sind per SOA eingebunden, ein gemeinsamer Web-Shop wird von allen Websites genutzt.

3 Aktuelle Bedrohungslage

3.1 Problemstellung

Die betrachteten Content Management Systeme sind komplexe Systeme mit vielen tausenden Zeilen Quellcode. Sie haben alle eine historische Entwicklung durchlaufen und sind von unterschiedlichsten Entwicklern gemeinsam erstellt worden. Unweigerlich sind Fehler (Schwachstellen) enthalten, die sich auf die Sicherheit der Websites auswirken können. Dies ist auch in zukünftigen Versionen der Software nicht zu vermeiden. Dabei stellt sich die Frage, ob die in der Vergangenheit gefundenen Schwachstellen Rückschlüsse auf zukünftig mögliche Fehler zulassen.

Vor die Beantwortung dieser Frage stellen sich mehrere Hürden, die hier auszugsweise beleuchtet werden.

Quellen: Zunächst gibt es unterschiedliche Quellen, die Schwachstellen auflisten. Wie sich unten zeigt, geben die Quellen unterschiedliche Zahlen wieder. Für die Open Source Projekte bestehen Meldungen zu den Schwachstellen, die intern behoben wurden, bevor sie als offizielle Schwachstellen bekannt wurden. Um die Informationen vergleichbar zu machen, werden die Daten zunächst kommentarlos aufgeführt und später im Fazit interpretiert.

Zeitraum: Eine weitere Entscheidung betrifft den Zeitraum, der betrachtet wird. Typisch für Softwareentwicklung ist es, dass Teile der Systeme über die Jahre neu geschrieben oder komplett überarbeitet werden. Um die „Reife“ eines Systems zu betrachten, sind die Teile des Systems wichtig, die in der aktuellen Version noch vorhanden sind. Es bringt folglich nichts, die Schwachstellen zu berücksichtigen, die vor zehn Jahren gefunden wurden. Selbst wenn bekannt ist, dass an einem Projekt die gleichen Entwickler arbeiten, ergäbe dies keinen Erkenntnisgewinn, da sich vermutlich gerade die Kernentwickler bei dem Thema Sicherheit weiterentwickelt haben. Wir beschränken uns deshalb auf den Zeitraum von Anfang 2010 bis Ende 2012.

Kern und Erweiterungen: Einige der Systeme, z.B. TYPO3, liegen in mehreren stabilen Release-Branches vor. Die offiziell gemeldeten Schwachstellen müssen auf die oben betrachtete Version nicht zutreffen. Andere Systeme wie Plone und Joomla! verwenden eine Plattform, die auch für andere Zwecke verwendet wird. So könnten zumindest Schwachstellen aufgeführt sein, die für das CMS nie relevant waren. Viele der Systeme arbeiten mit Erweiterungen, die deutlich weniger genutzt und getestet sind als der Kern des Systems. Wie oben beschrieben sind einige der CMS stark heterogen, was die Code Qualität zwischen CMS Kern und Erweiterungen/Plugins/Modulen anbelangt. Deshalb werden die Ergebnisse unten in Kern und Erweiterungen unterschieden.

Bekanntwerden: Insbesondere die Zeiten vom Bekanntwerden einer Schwachstelle bis zur Verfügbarkeit eines Patches sind interessant. Hier handelt es sich um brisantes Material, was deshalb nicht in statistisch auswertbarer Menge vorliegt.

Basistechnologien: Seitens der eingesetzten Basistechnologien (PHP, JAVA, CORBA etc.) können ebenso Schwachstellen auftreten. Hier trifft jedoch die Verallgemeinerung zu, dass Fehler in häufig genutztem Quellcode früher gefunden werden als in wenig genutztem Quellcode. Die Fehler, die erst gefunden werden, wenn der Quellcode bereits weithin in Benutzung ist, tauchen dann in den Common Vulnerabilities and Exposures (CVE) auf. Fehler in den Kernmodulen oder in den Basistechnologien fallen meist früher auf. Ungepatchte, veraltete Systeme stellen jedoch ein immer wieder anzutreffendes Risiko dar. Es werden jedoch auch fehlerhafte Updates veröffentlicht. Durch diese Lücken entstehen neue Szenarien, die zu neuen Sicherheitsrisiken führen.

Um die hier genannten Hürden zusammenzufassen: Wesentlich für die Beurteilung des Risikos ist das Wissen über die Eintrittswahrscheinlichkeit. Der Dienstanbieter muss sich darauf verlassen können, dass jeder „Vorfall“ gemeldet wird. In den folgenden Ausführungen wird die vorhandene Zahlenbasis ausgewertet, um eine Aussage über die Reife der Systeme aus sicherheitstechnischer Sicht wagen zu können.

3.2 Erhebung

3.2.1 Drupal

Das Drupal Security Team hat für den Drupal Kern für 2011 und 2012 größere Security Announcements herausgegeben, in denen es sich auf verschiedene CVE's bezieht[52]. Dabei schätzt das Team selber die Schwere der Schwachstellen im Kern für 2012 auf 3-5 von 5 möglichen Gefährdungsstufen. Die Schwachstellen sind nach eigener Aussage alle von außen ausnutzbar.[53]

Meldungen im „Security News" Archiv	***Anzahl in***		
	2010	***2011***	***2012***
Drupal Core	2	3	4
Drupal Contributed Projects (Module, Themes und Installationsprofile)	113	59	174

Tabelle 3.1: Drupal – Meldung aus "Security News" Archiv

Die Entwicklung der Meldungen der National Vulnerability Database (NVD)[54] und CVE Details für den „Vendor" Drupal weichen stark voneinander ab, allerdings ist für 2012 aus allen Quellen ein Anstieg der bekannten Schwachstellen zu verzeichnen (vgl. Tabelle 3.2).

Meldungen für Drupal von	***Anzahl in***		
	2010	***2011***	***2012***
cvedetails.com	8	3	13
nvd.nist.gov	51	10	163
portal.cert.dfn.de	11	8	15

Tabelle 3.2: Drupal – Schwachstellenmeldungen

Die Schwere der Schwachstellen für Drupal generell und seine Module wird über den CVSS Score[55] der CVE Details für den Zeitraum 2010-2012 wie folgt eingestuft:

52 https://www.drupal.org/security
53 http://drupal.org/security-team/risk-levels
54 NVD: http://web.nvd.nist.gov/
55 vgl. http://nvd.nist.gov/cvss.cfm

CVSS Score	*Anzahl der Schwachstellen*	*Anteile*
0-1	0	0,00%
1-2	0	0,00%
2-3	2	8,30%
3-4	3	12,50%
4-5	6	25,00%
5-6	9	37,50%
6-7	2	8,30%
7-8	2	8,30%
8-9	0	0,00%
9-10	0	0,00%
Total	24	
Gewichtetes Mittel CVSS Score: 5,5		

Tabelle 3.3: Drupal – CVSS Score[56]

Das bedeutet[57], dass die Schwachstellen des Drupal Systems unter Berücksichtigung aller Module im Durchschnitt die Bewertung „medium“ erhält.

Nach Aussagen im Drupal Security White Paper[58] und unter drupal.org[59] sind Schwachstellen in Erweiterungen weitaus häufiger als in Kern-Modulen der Basis-Installation zu finden, was sich mit den folgenden Werten für den Zeitraum 2010-2012 für alle Drupal Versionen aus der NVD deckt (vgl. Abbildung 3.1).

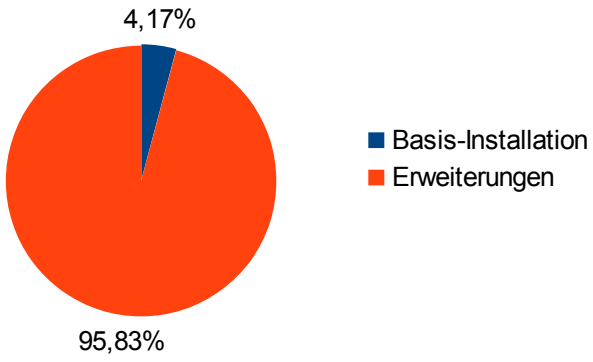


Abbildung 3.1: Drupal – Schwachstellen: Basis-Installation und Erweiterungen

Die Autoren des Drupal Security White Papers stellen dar, dass XSS Schwachstellen den Hauptanteil der Schwachstellen ausmachen, gefolgt von Access Bypass Schwachstellen. In CVE Details[60] sind zusätzlich noch „Execute Code“ Schwachstellen gelistet. Über ExploitDatabase[61] lassen sich derzeit 12 Suchergebnisse für

56 Tabelle wurde aus „CVE Details“ (http://www.cvedetails.com) entnommen
57 Nach http://nvd.nist.gov/cvss.cfm
58 http://drupalsecurityreport.org/sites/drupalsecurityreport.org/files/drupal-security-white-paper-1-1.pdf
59 http://drupal.org/documentation/is-drupal-secure
60 http://www.cvedetails.com/vendor/1367/Drupal.html
61 http://www.exploit-db.com

Drupal darstellen, dabei ist auch mindestens 1 Exploit für die aktuelle 7'er Version von Drupal öffentlich einsehbar.

Weiterhin ist mit Hilfe von secunia.com eine Auswertung möglich, die zeigt, welche der gemeldeten Schwachstellen vom Hersteller Drupal behoben wurden und welche nicht (vgl. Tabelle 3.4). Allerdings konnte beispielhaft nachvollzogen werden, dass die auf secunia noch als „unpatched“ gelisteten Schwachstellen bereits behoben wurden.

Schwachstellen für Drupal 7 im Zeitraum 2003-2012	*7 Schwachstellen*
Unpatched	14,00%
Vendor Patch	86,00%
Vendor Workaround	0,00%
Partial Fix	0,00%

Tabelle 3.4: Drupal – Schwachstellen

Das Drupal Security Team veröffentlicht Meldungen zu Schwachstellen und bietet diese Infos zyklisch an. Laut eigener Aussage stehen sie im engen Kontakt zu Kernentwicklern und Modul-Herstellern. Auf einige Schwachstellen reagieren die Sicherheitsexperten unabhängig von Release Zyklen. Im Rahmen einer Schwachstelle innerhalb des Zusatzmoduls „Context“ hat das Security Team von Drupal schon vor den Entwicklern des Moduls Workarounds bereitgestellt, um die Lücke zu schließen.[62]

3.2.2 Plone

Die Meldungen im Security Archiv von plone.org betreffen Schwachstellen in Kern-Komponenten von Plone sowie auch im Zope-Server (vgl. Tabelle 3.5). Das bisher einzige Advisory für 2012 im November behebt gleich 24 Schwachstellen auf einen Schlag[63].

Meldungen im „Security News“ Archiv	*Anzahl in*		
	2010	*2011*	*2012*
Plone und Zope	1	8	1

Tabelle 3.5: Plone – Meldung aus "Security News“ Archiv

Die Entwicklung von Schwachstellen-Meldungen 2010-2012 über cvedetails und NVD ergeben ein vergleichbares Ergebnis zu den Hersteller Seiten, wobei über Plone und Zope überhaupt kaum Meldungen zu finden sind (vgl. Tabelle 3.6).

Meldungen für Plone (und Zope)	*Anzahl in*		
	2010	*2011*	*2012*
cvedetails.com	1	9	-
nvd.nist.gov	1	9	0

Tabelle 3.6: Plone – Schwachstellenmeldungen

Die Schwere der Schachstellen generell für Plone und Zope wird über den CVSS Score der CVE Details für den Zeitraum 2010-2012 wie folgt eingestuft:

62 http://www.heise.de/newsticker/meldung/XSS-Luecke-in-Drupal-Modul-beseitigt-Update-997913.html)
63 http://plone.org/products/plone/security/advisories/20121106/

CVSS Score	*Anzahl der Schwachstellen*	*Anteile*
0-1		0,00%
1-2		0,00%
2-3		0,00%
3-4	1	10,00%
4-5	3	30,00%
5-6	2	20,00%
6-7		0,00%
7-8	2	20,00%
8-9		0,00%
9-10	2	20,00%
Total	10	
Gewichtetes Mittel CVSS Score: 6,7		

Tabelle 3.7: Plone – CVSS Score

Dabei werden 4 der 10 Lücken als XSS basierte Schwachstellen gekennzeichnet. Die beiden Schwachstellen des Schweregrades „high“ betreffen eine Code Execution Schwachstelle und eine unspezifizierte Schwachstelle in der CMF-Edition Komponente.

Die Auftrennung der Schwachstellen in Basis-Installation und Erweiterungen für den Zeitraum 2010-2012 stellt sich bei Plone schwierig dar, da die betroffenen Erweiterungen Teil der Basis-Installation sind. So ergibt sich nach Recherche über die NVD trotzdem ein Verhältnis wie in Abbildung 3.2 dargestellt[64].

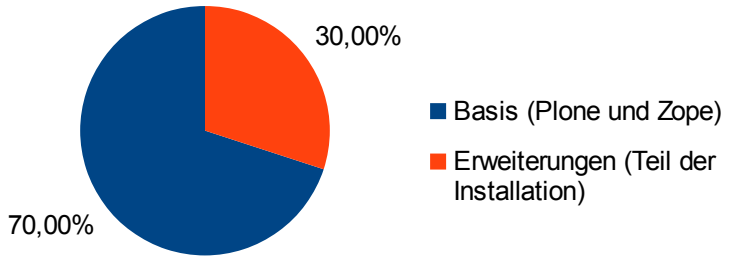


Abbildung 3.2: Plone – Schwachstellen: Basis-Installation und Erweiterungen

Über secunia.com ergab die Recherche für Plone 4, dass eine der sechs gemeldeten Schwachstellen nicht behoben wurde (vgl. Tabelle 3.8). Die Lücke hat den Schweregrad „less critical“, betrifft eine Hashtable-Funktion in Python generell, die für DoS Angriffe über Plone ausgenutzt werden kann[65]. Da insgesamt nur sechs Schwachstellen gelistet sind, führt das in folgender Tabelle zu 17% ungepatchten Schwachstellen, was natürlich dem Anliegen dieser Statistik nicht gerecht wird.

64 http://web.nvd.nist.gov/
65 http://secunia.com/advisories/47406/

Schwachstellen für Plone 4 im Zeitraum 2003-2012	*Anzahl der Schwachstellen*	*Schwachstellen in %*
Unpatched	1	17,00%
Vendor Patch	5	83,00%
Vendor Workaround	0	0,00%
Partial Fix	0	0,00%

Tabelle 3.8: Plone – Schwachstellen

Eine Recherche bei ExploitDatabase zeigt nur einen Eintrag zu Plone. Für einen Zeitrahmen zum Bereitstellen von Patches oder Bugfixes gibt es nur einen Anhaltspunkt. Der vorhin bereits erwähnte, am 6. November 2012 bereitgestellte Patch wurde am 31. Oktober 2012 offiziell gemeldet[66].

3.2.3 WordPress

Über wordpress.org gibt es in der Security Kategorie der Neuigkeiten Meldungen zu Schwachstellen. Die Meldungen betreffen lediglich Schwachstellen im Kernsystem von WordPress und umfassen jeweils mehrere gelöste Probleme[67] (vgl. Tabelle 3.9).

Das WordPress-Security Team behebt Probleme, die durch Externe und Kernentwickler gefunden wurden.

Meldungen im „Security News“ Archiv	*Anzahl in*		
	2010	*2011*	*2012*
WordPress	3	6	4

Tabelle 3.9: WordPress – Meldung aus "Security News“ Archiv

Die Entwicklung von Schwachstellen im Zeitraum 2010-2012 über die NVD und CVE-Details zeigt einen Anstieg der gemeldeten Schwachstellen (vgl. Tabelle 3.10).

Meldungen für WordPress	*Anzahl in*		
	2010	*2011*	*2012*
cvedetails.com	2	12	31
nvd.nist.gov	6	33	73
portal.cert.dfn.de	3	11	5

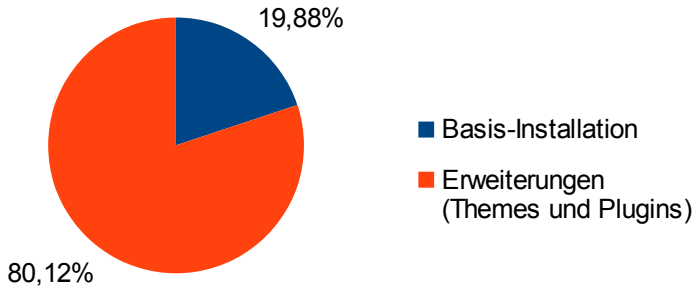
Tabelle 3.10: WordPress – Schwachstellenmeldungen

Über CVE Details sind 2012 allein 13 XSS basierte Schwachstellen gekennzeichnet. Die Schweregrade der Schwachstellen im Zeitraum 2010-2012 werden im Durchschnitt auf den Wert 6.1 und damit „medium“ eingestuft (vgl. Tabelle 3.11).

66 http://www.heise.de/security/meldung/Kritische-Luecken-in-Plone-und-Zope-1740907.html
67 https://wordpress.org/news/category/security/

CVSS Score	*Anzahl der Schwachstellen*	*Anteile*
0-1		0,00%
1-2		0,00%
2-3	4	8,70%
3-4	2	4,30%
4-5	13	28,30%
5-6	13	28,30%
6-7	5	10,90%
7-8	4	8,70%
8-9		0,00%
9-10	5	10,90%
Total	46	
Gewichtetes Mittel CVSS Score: 6,1		

Tabelle 3.11: WordPress – CVSS Score



Der Verteilung der gemeldeten Schwachstellen von der NVD auf die Basis-Installation und Erweiterungen für alle WordPress Versionen im Zeitraum 2010-2012 wurde in der Abbildung 3.3 dargestellt[68].

Eine Analyse der Schwachstellen-Meldungen der secunia.com zeigt für WordPress in allen 3.X'er Versionen zwei ungepatchte Schwachstellen vom Hersteller (vgl. Tabelle 3.12). Dabei handelt es sich um eine CSRF-Schwachstelle und um einen Schwachstelle für das Erraten von validen Nutzerkonten.[69]

68 http://web.nvd.nist.gov/
69 http://secunia.com/advisories/product/33191/?task=advisories

Schwachstellen für WordPress 3.X im Zeitraum 2003-2012	*Anzahl der Schwachstellen*	*Schwachstellen in %*
Unpatched	2	14,00%
Vendor Patch	12	86,00%
Vendor Workaround	0	0,00%
Partial Fix	0	0,00%

Tabelle 3.12: WordPress – Schwachstellen

Eine Suche nach Exploits mit Hilfe von ExploitsDatabase ergab über 250 Exploits für WordPress. Hier ist der Großteil der Exploits für WordPress Plugins eingetragen worden.

3.2.4 Joomla!

Joomla! hat für die Behandlung von Schwachstellen ein „Security Strike Team“ aufgestellt, welches die aktuelle Joomla! Version aktiv betreut. Vom Team wurden 2012 für mehr als 15 Schwachstellen aller Schweregrade die dazu gehörigen Patches zur Verfügung gestellt (vgl. Tabelle 3.13). Die Meldungen im Security News Archiv der Developer Community von Joomla! betreffen keine Erweiterungsmodule, sondern nur Schwachstellen in Kern-Komponenten[70] (vgl. Tabelle 3.13)

Meldungen im „Security News“ Archiv	*Anzahl in*		
	2010	*2011*	*2012*
Joomla! Core	11	35	22

Tabelle 3.13: Joomla! – Meldung aus "Security News“ Archiv

Ein klarer Trend über die Schwachstellen-Meldungen der NVD und CVE Details ist über die letzten drei Jahre nicht zu erkennen (vgl. Tabelle 3.14). Feststellbar ist, das XSS basierte Schwachstellen statistisch zugenommen haben und Execute Code und SQL Injection Schwachstellen-Meldungen abnehmen[71].

Meldungen für Joomla!	*Anzahl in*		
	2010	*2011*	*2012*
cvedetails.com	10	16	25
nvd.nist.gov	52	22	35
portal.cert.dfn.de	-	-	-

Tabelle 3.14: Joomla! – Schwachstellenmeldungen

Bei der Verteilung von Schwachstellen zeigt sich vergleichbar zu Drupal und WordPress, dass die Kern-Komponenten des Systems weit weniger Schwachstellen aufweisen als die Module von Drittanbietern (vgl. Abbildung 3.4).[72]

70 http://developer.joomla.org/security.html
71 http://www.cvedetails.com/vendor/3496/Joomla.html
72 http://web.nvd.nist.gov/

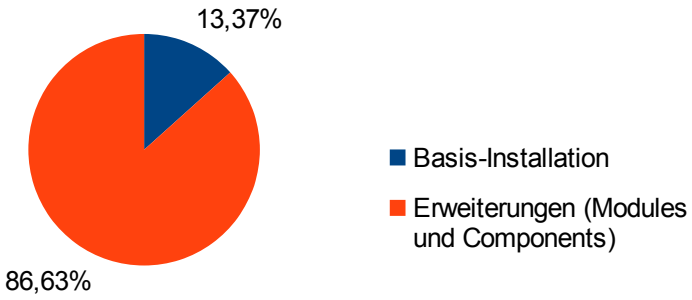


Abbildung 3.4: Joomla! – Schwachstellen: Basis-Installation und Erweiterungen

Die Schwere der Schwachstellen über CVE Details generell für den Joomla!-Kern und alle Module wird für den Zeitraum 2010-2012 über den CVSS Score wie in Tabelle 3.15 dargestellt eingestuft.

CVSS Score	*Anzahl der Schwachstellen*	*Anteile*
0-1	0	0,00%
1-2	0	0,00%
2-3	0	0,00%
3-4	1	2,00%
4-5	19	37,30%
5-6	16	31,40%
6-7	1	2,00%
7-8	14	27,50%
8-9	0	0,00%
9-10	0	0,00%
Total	51	
Gewichtetes Mittel CVSS Score: 6,2		

Tabelle 3.15: Joomla! – CVSS Score

Damit liegt Joomla!, nach NVD, bei der Berücksichtigung aller Module im Durchschnitt bei der Bewertung „medium“.

Laut Secunia sind alle Schwachstellen für Joomla! 3 durch einen Patch des Herstellers beseitigt worden. Die Statistik ist hier zu vernachlässigen, weil lediglich eine Schwachstelle betrachtet wurde. Auffällig und wichtig zur Bedrohungslage ist bei Joomla!, dass unabhängig von der Version und dem Zeitraum bei ExploitDatabase über 800 Einträge zu Joomla! und deren Modulen gelistet werden, wobei hier erneut Module von Drittanbietern die größte Angriffsfläche bieten. Auch bei der Ende 2012 erschienenen Meldung bei heise Security, dass durch Joomla! verwaltete Websites zur Verbreitung von Malware genutzt werden könnten, war ein Zusatzmodule (JCE Editor) Ursache der Schwachstelle[73].

73 http://www.heise.de/security/meldung/Joomla-Seiten-als-Malware-Schleudern-missbraucht-1766717.html

3.2.5 TYPO3

Das TYPO3 Security Team erstellt „Security Announcements“. Für den Zeitraum 2010-2012 ist feststellbar, dass die Schwachstellen im Kernbereich von TYPO3 gegenüber der gelisteten Schwachstellen in Erweiterungsmodulen lediglich rund ein Drittel betragen[74] (vgl. Tabelle 3.16).

Security Advisories	***Anzahl in***		
	2010	***2011***	***2012***
TYPO3 Core	6	4	5
TYPO3 Extensions	16	18	13

Tabelle 3.16: TYPO3 – Security Advisories

Unter Einbeziehung der drei Quellen zu CVE's ist zumindest kein klarer Trend sichtbar. Innerhalb der Schwachstellen-Entwicklung sind auf CVE Details vor allem Code Execution, SQL Injection und XSS Schwachstellen zu verzeichnen.

Meldungen für TYPO3	***Anzahl in***		
	2010	***2011***	***2012***
cvedetails.com	26	-	31
nvd.nist.gov	70	15	42
portal.cert.dfn.de	17	24	18

Tabelle 3.17: TYPO3 – Schwachstellenmeldungen

Die Schweregrade der Schwachstellen über CVE Details in diesem Zeitraum werden im Durchschnitt auf den Wert 6 und damit „medium“ eingestuft. Allerdings werden 28 Prozent der Schwachstellen mit dem Schweregrad „high“ gekennzeichnet (vgl. Tabelle 3.18).

74 http://typo3.org/teams/security/security-bulletins

CVSS Score	*Anzahl der Schwachstellen*	*Anteile*
0-1	0	0,00%
1-2	0	0,00%
2-3	1	1,80%
3-4	7	12,30%
4-5	17	29,80%
5-6	12	21,10%
6-7	4	7,00%
7-8	16	28,10%
8-9	0	0,00%
9-10	0	0,00%
Total	57	
Gewichtetes Mittel CVSS Score: 6,0		

Tabelle 3.18: TYPO3 – CVSS Score

Die Trennung der Schwachstellen nach Basis-Installation und Erweiterungen für den Zeitraum 2010-2012 aller Versionen von TYPO3 zeigt einen noch deutlicheren Anteil der Erweiterungen an Schwachstellen als die vom TYPO3 Security Team erstellte Verteilung der „Security Announcements“. Auch hier ist die Verteilung ähnlich wie bei WordPress, Drupal und Joomla! (vgl. Abbildung 3.5).

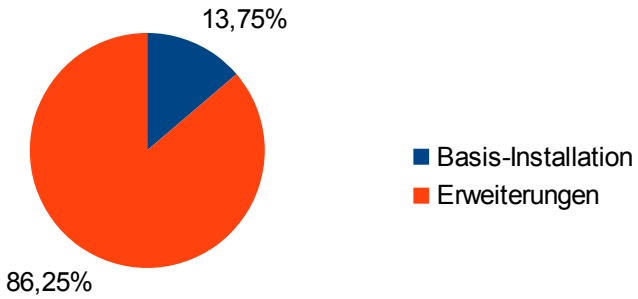


Abbildung 3.5: TYPO3 – Schwachstellen: Basis-Installation und Erweiterungen

Weiterhin ist mit Hilfe von secunia eine Auswertung für TYPO3 in den Versionen 4.X möglich, welche gemeldeten Schwachstellen im Zeitraum 2003-2012 behoben wurden. Für alle Schwachstellen wird laut dieser Statistik ein Patch angeboten (vgl. Tabelle 3.19).

Schwachstellen für WordTYPO3 4.X im Zeitraum 2003-2012	*23 Schwachstellen*
Unpatched	0,00%
Vendor Patch	100,00%
Vendor Workaround	0,00%
Partial Fix	0,00%

Tabelle 3.19: TYPO3 – Schwachstellen

Über ExpoitsDatabase lassen sich momentan ca. neun Exploits anzeigen, für die aktuelle 4er Version allerdings nur zwei Exploits.

3.2.6 Basis-Technologien

Die betrachteten CMS sind in unterschiedlichen Programmiersprachen entwickelt. Die Sprachen werden in komplexen Anwendungsentwicklungsumgebungen ausgeliefert, zu denen umfangreiche Bibliotheken mitgeliefert werden. Einige der darin gefundenen Schwachstellen können unter Umständen über das CMS ausgenutzt werden (vgl. Tabelle 3.20).

Entwicklung von Schwachstellen Basis-Technologien	*2010*	*2011*	*2012*
PHP	35	35	22
Python	7	3	7
Java JRE	54	57	45

Tabelle 3.20: Entwicklung der Schwachstellen in verwendeten Basistechnologien[75]

2012 war das Jahr der Java Schwachstelle. Im Oktober 2012 schloss Oracle insgesamt 30 Schwachstellen in Java[76], allein 10 davon haben den höchsten CVSS Schweregrad. Untersuchungen zeigen, dass gerade Java Schwachstellen im Jahr 2012 sehr häufig über Exploits ausgenutzt wurden[77]. Betroffen sind hier im Großteil nur die Java Installation auf Seiten des Clients, nur 2 der 30 Schwachstellen des Oktober Patch Releases sind auch auf Java Server Versionen anwendbar.

Bei PHP war zum Beispiel eine Schwachstelle in der XML-RPC Bibliothek von PHP für alle PHP basierten CMS-Systeme von Bedeutung, da diese Bibliothek verwendet wird. Ebenso ermöglichte ein JQuery Plugin „Uploadify“ auf der Basis von PHP mehrere Schwachstellen in WordPress Erweiterungen[78].

Die aufgeführten Schwachstellen müssen in jedem Fall einzeln auf ihre Relevanz im Serverumfeld geprüft werden. Auch wenn es relative Unterschiede zwischen den Sprachen gibt, auf die hier nicht eingegangen wird, lassen die insgesamt niedrigen Zahlen den Schluss zu, dass alle Sprachen bzgl. der offenbarten Sicherheitslücken ähnlich gut für CMS geeignet sind. Die über den betrachteten Zeitraum erfassten, extrem niedrigen Zahlen für Python sind ein Anzeichen für die sehr gute Reife dieser Sprache.

75 http://www.cvedetails.com
76 http://www.oracle.com/technetwork/topics/security/javacpuoct2012-1515924.html
77 http://www.securelist.com/en/analysis/204792250/IT_Threat_Evolution_Q3_2012#4
78 http://www.heise.de/security/meldung/WordPress-Modul-Uploadify-als-Einfallstor-1625729.html

3.2.7 Datenbank Systeme

Die Entwicklung der Schwachstellen in verwendeten Datenbanken[79] wurde in der Tabelle 3.21 dargestellt.

Datenbank-System	*2010*	*2011*	*2012*
MySQL	6	16	59
Oracle Database Server	31	49	24
PostgreSQL	7	1	9
Microsoft SQL Server	-	1	3

Tabelle 3.21: Entwicklung von Schwachstellen Datenbank-Systemen

Bei der zu nutzenden Datenbank lassen mehrere der betrachtenden CMS dem Nutzer die Wahl zwischen verschiedenen Systemen. Auch hier gilt, dass die Schwachstellen der DB Systeme nicht sofort über das CMS ausgenutzt werden können. Es ist eine Analyse der konkreten Schwachstelle erforderlich. Zumindest ist zu unterscheiden, ob es sich um Schwachstellen des Datenbankservers, seiner Administrationswerkzeuge oder verwendeter Client-Module handelt.

3.3 Auswertung

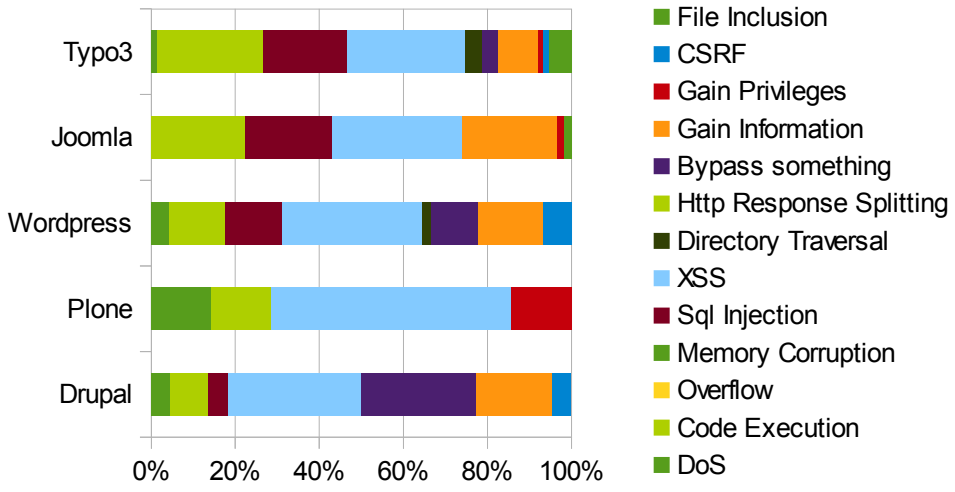
3.3.1 Klassifizierung nach Schwachstellentypen

Eine Auswertung der für die CMS gemeldeten Schwachstellen im Zeitraum 2010-2012[80] zeigt die Verteilung der gemeldeten Schwachstellen auf verschiedene Angriffsvektoren (vgl. Abbildung 3.6).
Einzelne Ergebnisse aus dieser Darstellung erklären sich schon aus der Architektur: das System Plone ist nicht anfällig gegen SQL-Injections, da schlicht keine SQL Datenbank verwendet wird.

FAKT: Die Anteile der Code-Execution Schwachstellen – die schwerwiegendsten Fehler – sind bei Joomla! und TYPO3 recht hoch.

Da dies PHP-Systeme sind, scheint der Vergleich zu Drupal und WordPress angemessen.

79 http://www.cvedetails.com
80 http://www.cvedetails.com



Die folgende Betrachtung vergleicht die Durchschnittswerte aller CMS bzgl. der genannten Schwachstellentypen.

	Dos	*Code Execution*	*Overflow*	*Memory Corruption*	*SQL Injection*	*XSS*	*Directory Traversal*	*HTTP Response Splitting*	*Bypass Something*	*Gain Information*	*Gain Privileges*	*CSRF*	*File Inclusion*
Durchschnitt	5	41	0	0	34	65	4	0	14	31	3	5	5

Tabelle 3.22: Durchschnittswerte aller CMS bzgl. der genannten Schwachstellentypen

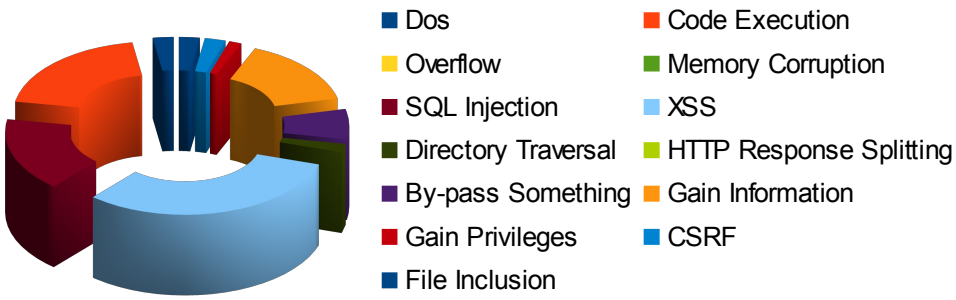


Abbildung 3.7: Graphische Darstellung der Durchschnittswerte aller CMS bzgl. der genannten Schwachstellentypen

Als Ergebnis kann festgestellt werden, dass XSS, Code Execution und SQL-Injection die drei häufigsten Schwachstellen darstellen.

3.3.2 Verteilung der Schwachstellen zwischen Basis-Installation und Erweiterung

FAKT: Anhand der Auswertung der Daten zeigt sich klar, dass ein Großteil der Schwachstellen in den Erweiterungen der CMS zu finden sind.

	Drupal	*Plone*	*WordPress*	*Joomla!*	*TYPO3*
Basis-Installation	4,17%	70,00%	19,88%	13,37%	13,75%
Erweiterungen	95,83%	30,00%	80,12%	86,63%	86,25%

Tabelle 3.23: Schwachstellen in den Erweiterungen und Basis-Installation

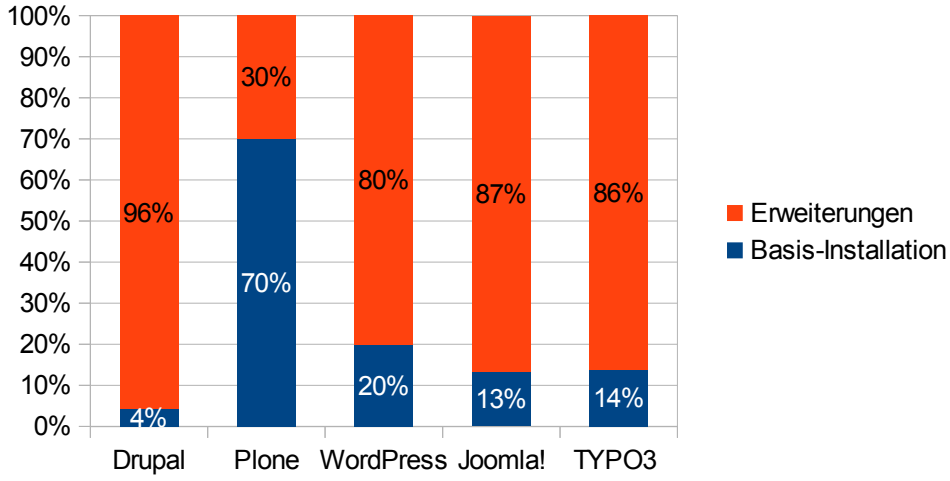


Abbildung 3.8: Graphische Darstellung der Schwachstellen in den Erweiterungen und Basis-Installation

Bei Plone waren die wenigen gefundenen Schwachstellen in Erweiterungsmodulen, die Teil der Basis-installation sind, deshalb ergibt sich die im Vergleich zu den anderen Systemen inverse Darstellung. Die Studie der IBM X-Force „2012 Mid-year Trend and Risk Report“[81] kommt zu einem vergleichbaren Ergebnis: das Verhältnis der Anzahl von Schwachstellen in Kern-Komponenten zur Anzahl der Schwachstellen in Erweiterungen beträgt dort eins zu fünf.

3.3.3 Schwachstellen-Kompensation

Zusätzlich fällt auf, dass im ersten Halbjahr 2012 Schwachstellen in Kern-Komponenten häufiger gepatcht werden als Schwachstellen in Erweiterungen. Das ist ebenso für 2011 der Fall (vgl. Tabelle 3.24).

81 http://www-03.ibm.com/security/xforce/downloads.html

	2011	*2012 H1*
CMS Core Schwachstellen patched	80,00%	70,00%
CMS Plug-in Schwachstellen patched	52,00%	52,00%

Tabelle 3.24: Behandlung der Schwachstellen

Die Studie81 wertet mit Secunia[82] Daten von 2003 bis 2012 aus und stellt fest, dass für alle betrachteten CMS über 80 Prozent aller Schwachstellen über einen Hersteller-Patch geschlossen wurden. Die Auswertung der Schweregrade der gemeldeten Schwachstellen zeigt, dass sich die zu betrachtenden CMS hier kaum unterscheiden (vgl. Tabelle 3.25).

	Drupal	*Plone*	*WordPress*	*Joomla!*	*TYPO3*
Durchschnittlicher CVSS Score	5,5	6,7	6,1	6,2	6

Tabelle 3.25: Schweregrad der Schwachstellen

3.3.4 Vergleich der CMS zueinander

Eine Auswertung der für die CMS gemeldeten Schwachstellen im Zeitraum 2010-2012[83] zeigt, dass für Plone die wenigsten und für TYPO3 die meisten Schwachstellen gefunden wurden.

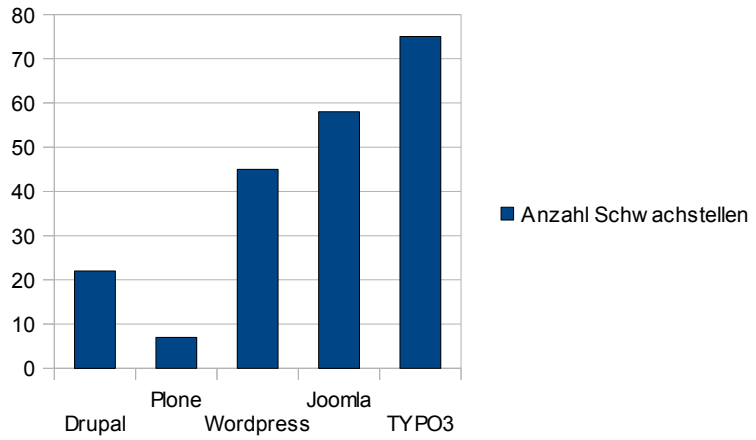


Abbildung 3.9: Schwachstellen absolut pro CMS

3.3.5 Diskussion

Die hier ausgewerteten Quellen sind aus den eingangs geschilderten Gründen als Anhaltspunkt, nicht aber als Kriterium für die Auswahl des CMS geeignet. Dennoch lassen sich einige Aussagen ableiten:

Transparenz: Alle Open Source CMS stellen eine Transparenz bzgl. ihrer Schwachstellen her. Der Dienstanbieter kann aufgrund dieser Informationen eine qualifizierte Entscheidung treffen, in dem er bezogen auf sein Anwendungsszenario prüft, welche Schwachstellen ihn in der Vergangenheit „überrascht“ hätten. Er kann prüfen oder prüfen lassen, wie das Security Team auf diese Schwachstelle reagiert hat, wie die Qualität

82 http://www.secunia.com
83 www.cvedetails.com

der Kommunikation in Bezug auf die Sicherheitslücken war und welche zusätzlichen Schwachstellen das Team selbst beseitigt hat.

Schwachstellen im CMS-Kern: Die Anzahl der im CMS Kern gefundenen Schwachstellen ist gering im Verhältnis zu den Schwachstellen in den Erweiterungen. Im Fall von Plone führt dieser simple Vergleich zu einem größeren Verhältnis, da bei Plone schon viele Erweiterungen im Kern enthalten sind. Die geringen absoluten Zahlen gefundener Schwachstellen bei Plone sprechen jedoch für die Qualität dieses CMS. Im Allgemeinen lassen die Ergebnisse jedoch nicht die Schlussfolgerung zu, dass die Beschränkung auf die Standardinstallation ausreichend sicher wäre.

Gefundene Schwachstellen: Die Anzahl der gefundenen Schwachstellen befreit keinen Dienstanbieter davon – egal welches CMS er wählt – permanent die Meldungen der Security-Teams zu verfolgen und darauf vorbereitet zu sein, ein Security Update schnellstmöglich einspielen zu müssen.

EMPFEHLUNG: Dienstanbieter müssen permanent in der Lage sein, Patches einzuspielen.

Auch bietet einzig die Auswertung der Anzahl von Schwachstellen-Meldungen keine belastbare Aussage. Die Anzahl der Meldungen ist abhängig vom Verbreitungsgrad der Software, der Anzahl der eingesetzten Module und natürlich auch der Anfälligkeit der zugrunde liegenden Basis-Technologien.

Ausnutzung der Schwachstellen: Die Häufigkeit von ausnutzbaren Schwachstellen in den CMS Systemen selbst muss für die Dienstanbieter Anlass genug sein, über eine Strategie der „Verteidigung in der Tiefe“ (defense in depth) nachzudenken. Die mit den CMS verbundenen Infrastrukturen müssen gegenseitig voreinander abgegrenzt und geschützt werden.

EMPFEHLUNG: Dienstanbieter sollten ihre Websites konzipieren, bevor sie sie aufsetzen. Das Prinzip „Verteidigung in der Tiefe“ ist dabei von herausragender Bedeutung.

Die Wahrscheinlichkeit von „Zero-Day-Exploits“ ist gegeben. Ein aktives Monitoring der Website kann dabei helfen, solche Situationen zu erkennen, bevor eine Schwachstelle vom CMS Hersteller veröffentlicht wird.

EMPFEHLUNG: Dienstanbieter sollten ihre Websites permanent monitoren.

Es ist positiv zu erwähnen, dass die Informationen zu auftretenden Schwachstellen bei allen Open Source Systemen transparent und nachvollziehbar sind.

4 Sicherheitsuntersuchung

4.1 Vorgehen

In dieser Phase der Untersuchung werden die sicherheitsrelevanten Aspekte der ausgewählten Content Management Systeme nach den in Kapitel 2.3 aufgestellten Kriterien bewertet.

Dazu wurden für die Open-Source-CMS die öffentlich zugänglichen Informationen wie die Systemdokumentation, Tutorials und Guidelines, aber auch Einträge in Foren oder Communities ausgewertet.

Im Rahmen dieser Studie waren die zur Verfügung stehenden Ressourcen begrenzt; sie soll eine erste Einschätzung der zu erwartenden Sicherheit beim Einsatz der betrachteten CMS ermöglichen. Daher konnten nicht alle Kriterien für alle Systeme mit gleicher Gründlichkeit untersucht werden. Wenn eine Information in den verfügbaren Quellen nicht binnen 15 Minuten[84] auffindbar war, wurde die Auswertung für das jeweilige CMS an dieser Stelle abgebrochen und dies in den Ergebnissen vermerkt. Die Tatsache, dass eine Information nicht (leicht) zu finden ist, gibt einem potenziellen Anwender eines Systems aber auch schon einen Hinweis.

Eine Übersicht der detaillierten Untersuchungsergebnisse für jedes untersuchte System findet sich im Anhang (vgl. ab Seite 99 ff). Hier finden sich auch, sofern verfügbar, entsprechende Links für weiterführende Informationen zum jeweiligen System und Kriterium. Einige Funktionalitäten bzw. Kriterien konnten im Rahmen dieser Studie nicht nachgewiesen bzw. geprüft werden (vgl. oben). Diese sind durch „np“ (not proved) gekennzeichnet. Andere Funktionalitäten konnten in angemessener Zeit (s.o., 15 Minuten Festlegung) in den Quellen nicht gefunden werden, sind aber möglicherweise doch für das jeweilige System verfügbar. Diese wurden durch „nf“ (not found) gekennzeichnet.

Vor einer Entscheidung für ein bestimmtes System sollten alle Informationen anhand der hier gegebenen Kriterien ausgewertet werden und ggf. durch eigene Kriterien und Gewichtungen ergänzt werden.

Dieses Kapitel beschreibt für jede der in Kapitel 2.3 zusammengestellten Phasen (Design, Transition und Operation) die Ergebnisse der Funktions- und Sicherheitsuntersuchung. Dabei werden für jedes Kriterium die Einzelergebnisse genannt, es wird evtl. auf Besonderheiten einzelner CMS hingewiesen und für die jeweilige Kriteriengruppe zusammengefasst.

4.2 Service Design

4.2.1 Geschäftsführung (Governance)

4.2.1.1 Lizenzierung: Ausprobierbarkeit

Da bei allen untersuchten Open-Source-Systemen die Software kostenlos zur Verfügung steht, steht hier immer die Möglichkeit einer eigenen Testinstallation offen. TYPO3 und Joomla! bieten hierfür spezielle Pakete an, mit denen mit wenig Aufwand eine lauffähige Instanz mit Beispielinhalten aufgesetzt werden kann.

Darüber hinaus existieren für alle Open-Source-CMS öffentliche Demo-Instanzen, die ein Ausprobieren ohne eigenen Server erlauben. Die öffentlich zugänglichen Demo-Instanzen erlauben es jedem, sich mit verschiedenen Berechtigungsstufen am Backend anzumelden und so Funktion und Bedienung auszu-

84 15 Minuten wurden als die Zeit angenommen, die wahrscheinlich ein Administrator maximal für die Suche verwenden würde.

probieren. Im Falle von TYPO3, Joomla! und Plone werden Demo-Instanzen vom jeweiligen Projekt selbst bereitgestellt.

Der Drittanbieter OpenSource CMS[85] betreibt öffentliche Demo-Instanzen von mehreren hundert CMS, darunter TYPO3, WordPress und Drupal. Zu beachten ist, dass bei öffentlichen Demo-Instanzen aus Sicherheitsgründen meistens nicht alle Funktionen zur Verfügung stehen.

4.2.1.2 Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen

Alle Systeme bieten ein umfangreiches Sortiment an Dokumentationen, Anleitungen und Schulungen, die öffentlich zugänglich sind. Die Dokumentationen scheinen auf den ersten Blick auch recht aktuell zu sein, allerdings sind bei TYPO3 einige veraltete Dokumentationen und Anleitungen für Entwickler aufgefunden worden (vgl. Tabellen ab Seite 99).

4.2.1.3 Transparenz der Kommunikation bei Schwachstellen

Bei allen Open-Source-CMS ist eine Kontaktadresse zur Meldung von Sicherheitsproblemen gut auffindbar und diese Projekte betreiben ein öffentliches Bug-Tracking-System, so dass Anwender den Fortgang der Behebung von Fehlern in der Software verfolgen können. Während der Behebung von Sicherheitslücken würde ein Bug-Tracking-System allerdings auch Angreifern Informationen über offene Schwachstellen liefern. Daher bietet keines der betrachteten CMS eine direkte Suchmöglichkeit für sicherheitsrelevante Fehler, die noch nicht behoben wurden, im Bug-Tracking-System an. Weitere Details hierzu wie bspw. Adressen sind in den Tabellen zum jeweiligen CMS im Anhang ab Seite 99 zu finden.

Sicherheitsrelevante Informationen (Advisories) sind bei allen betrachteten Open-Source-Systemen leicht auffindbar und meist gut strukturiert. Auch hier sei auf die Tabellen im Anhang ab Seite 99 verwiesen.

Den Prozess zum Umgang mit Sicherheitslücken machen die Open Source Projekte mit Ausnahme von Joomla! öffentlich.

4.2.1.4 Sicherer Entwicklungsprozess

Schlecht strukturierter, schwer wartbarer oder missverständlicher Quelltext ist eine potenzielle Fehlerquelle und Ursache für Sicherheitslücken. Bei Open-Source-Projekten mit häufig wechselnden, nicht-hauptamtlichen Entwicklern ist deshalb Sorgfalt besonders wichtig.

Für alle betrachteten Open-Source CMS konnten öffentliche Richtlinien und Qualitätskriterien für Quelltexte gefunden werden. Auch die Kommentierung der Quelltexte ist ausführlich und für mit dem Projekt vertraute Entwickler leicht verständlich. Organisatorische Maßnahmen, wie Review-Prozesse, sind für die Open-Source-CMS mit Ausnahme von Joomla! öffentlich dokumentiert. Alle Projekte betreiben ein (nicht immer vollständig öffentliches, s. Joomla!) Bug-Tracking-System mit dem die Behebung von Fehlern und die Implementierung neuer Funktionen verfolgt und koordiniert werden kann.

4.2.1.5 Dokumentation der Sicherheitsanforderungen

Die Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen und Einschränkungen ermöglicht dem Dienstanbieter, seine Anforderungen mit den Spezifikationen des CMS einfach zu vergleichen. Diese Dokumentation ist für alle Systeme mit Ausnahme von Joomla! verfügbar.

85 http://www.opensourcecms.com

4.2.1.6 Beschreibung aller Sicherheitsmechanismen

Wenn ein externes Sicherheitsreview durchgeführt werden soll, ist die Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design-, und Implementierungsebene sehr hilfreich. Leider sind diese Dokumente nur für Plone verfügbar. Das Dokument „Security overview of Plone“[86] nennt die gängigen Sicherheitsprobleme und beschreibt, wie diese in Plone behandelt werden. Das Plone-Entwicklerhandbuch[87] eines Drittanbieters erläutert im Abschnitt Sicherheit und Arbeitsabläufe detailliert Sicherheitsmechanismen und ihre Anwendung anhand eines konkreten Beispiels. Bei den übrigen Systemen muss man sich mittels unterschiedlicher Dokumente einen Überblick verschaffen. Weitere Details hierzu finden sich in den Tabellen unter dem Kriterium „Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene“ im Anhang ab Seite 99.

4.2.2 Prüfung / Test (Verifikation)

Für die Stabilität, Robustheit und Fehlerfreiheit eines Softwareprodukts sind während der Entwicklung häufige Überprüfungen des Designs und der entstandenen Programmelemente von hoher Bedeutung.

Auch erfahrene Software-Architekten und Entwickler machen Fehler und nicht jeder Entwickler ist gleichzeitig ein Sicherheitsexperte. Um sicherheitsrelevante Fehler aufzudecken, ist es sinnvoll, einen Review von Experten durchführen zu lassen. Besonders wichtig sind Prüfungen des Software-Designs sowie der Implementierung auf Schwächen. Besonders gründlich und aufschlussreich, aber auch aufwändig ist eine Analyse des Quelltextes.

Leider ist nur wenig über durchgeführte Sicherheitsanalysen der betrachteten CMS veröffentlicht. Bei Open-Source Projekten, die ein explizites „Security Team“ haben (TYPO3, WordPress, Joomla!, Drupal und Plone), kann davon ausgegangen werden, dass intern eine Prüfung stattfindet. TYPO3, Drupal und Plone haben zumindest mehr oder weniger formale Richtlinien zur Prüfung von externen Erweiterungen vor deren Aufnahme in die offizielle Liste. Dabei werden Kernkomponenten von TYPO3 nach Änderungen immer ausgiebigen Tests unterzogen, Erweiterungen werden zumindest einem Review unterzogen, wenn schon mehrfach Fehler festgestellt worden sind. Bei Drupal werden neue und geänderte Kern- und Erweiterungskomponenten sowohl einem automatisierten Test unterzogen als auch durch ein Community Member begutachtet. Plone hat einen detaillierten Workflow für das Einreichen von neuen Komponenten oder Bug Fixes, in dessen Rahmen auch größtenteils automatisierte Tests vorgesehen sind. Alle diese Tests betreffen aber in erster Linie die Funktionalität der neuen oder geänderten Module; inwieweit Tests oder Reviewer Sicherheitsaspekte betrachten, bleibt offen.

4.2.3 Softwareentwicklung (Construction)

4.2.3.1 Sichere Architektur: Modularisierung

Die Modularisierung und Trennung der logischen Funktionen wie Auslieferung, Nutzerverwaltung, Content Verwaltung, Datenhaltung (vgl. Kapitel 2.1), ein Grundprinzip der Softwareentwicklung, wird von allen Systemen grundsätzlich umgesetzt. Alle Open Source Systeme realisieren dies auf Komponentenebene. Die PHP basierten Systeme TYPO3, WordPress, Joomla! und Drupal benutzen darüber hinaus das LAMP- bzw. WAMP Prinzip (Linux/Windows, Apache, MySQL, PHP). Die einzelnen Module können bei Bedarf mit gleichwertiger Software ersetzt werden. Die Funktionen sind entsprechend logisch auf die Module aufgeteilt. Plone bietet aufgrund seiner Architektur, die auf dem Applikationsserver Zope basiert, auch eine flexible Modularisierung (vgl. Kapitel 2.2.2.1 und 2.2.2.2).

86 http://plone.org/products/plone/security/overview
87 http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch

Grundsätzlich sollten auch vertrauenswürdige und nicht vertrauenswürdige Systembestandteile architektonisch voneinander getrennt sein bspw. durch Sandboxing. Ein „Sandboxing“, wie es bspw. bei Java zu finden ist, gibt es für PHP-Systeme nicht. Für Python steht grundsätzlich der „RestrictedPython“ – Mode zur Verfügung, der aber für das aktuelle Plone nicht gepflegt ist und nicht verwendet werden kann. So müssen alle nicht-Java-Systeme auf Betriebssystem-Mittel zur Kapselung der Komponenten zurückgreifen.

4.2.3.2 Sichere Architektur: Basistechnologie

Die Verwendung von aktuellen, sicheren Bibliotheken ist ein wesentlicher Aspekt in der Softwareentwicklung.

Die Verwendung häufig eingesetzter Bibliotheken wie bspw. jquery helfen Sicherheitslücken – oft verursacht durch eigene proprietäre Lösungen – zu vermeiden. Da verbreitete und häufig verwendete Bibliotheken nicht nur mehr Sicherheit bieten, sondern auch die Entwicklung wesentlich vereinfachen, werden diese auch von allen Systemen, in Abhängigkeit von der jeweiligen Plattform, grundsätzlich verwendet.

Die CMS TYPO3, WordPress, Joomla! und Drupal bauen als PHP-basierte Systeme auf den Komponenten Webserver, Datenbank und PHP-Interpreter auf. Für die Aktualisierung dieser Software und der durch sie verwendeten Bibliotheken ist der Betreiber des Servers verantwortlich, der dafür sorgen muss, dass jeweils die aktuellsten (kompatiblen) Versionen von PHP auf dem Server installiert sind.

Plone bringt mit dem „Unified Installer“ alle benötigten Komponenten mit und ändert sonst nichts auf dem Server, sofern die Voraussetzungen wie insbesondere die GNU Compiler Collection vorinstalliert sind. Die hierfür zu installierenden Pakete können je nach Linux-Distribution ein wenig variieren. Installiert werden dann im Wesentlichen die zur jeweiligen Plone-Version passende Python-Version sowie das Zope 2 Framework und natürlich auch Plone selbst.

Da alle hier betrachten CMS keine „eigenen“ Bibliotheken auf dem Server installieren oder aktualisieren, liegt es in erster Linie am Administrator der Site, dass die Standardbibliotheken auf einem aktuellen Stand gehalten werden bzw. notwendige Sicherheitspatches eingespielt werden, sofern dies nicht automatisch geschieht.

4.2.3.3 Sichere Architektur: Integrationsfähigkeit

Die Integrationsfähigkeit eines CMS kann durch die Verwendung von Standards wie Web Services, Paymentstandards sowie standardisierter Verfahren zu Authentisierung und Autorisierung unterstützt werden.

Web Services werden bei allen Systemen, wenn überhaupt, nur rudimentär unterstützt. Es gibt in der Realisierung Unterschiede, so erlaubt Plone (mit Programmieraufwand) zum Beispiel eine Anbindung über Python und Drupal stellt mehrere Erweiterungen für Services und Clients bereit. Im Rahmen dieser Studie konnte jedoch die Sicherheit und Funktionsfähigkeit dieser Schnittstellen/Erweiterungen nicht näher geprüft werden. Grundsätzlich gilt für alle Systeme, dass Web Services Schnittstellen – falls implementiert – nicht gegen WS-I[88] Profile bzgl. Interoperabilität und Sicherheit getestet sind. Deshalb sollte ein Dienstanbieter davon ausgehen, dass sie nur durch versierte, programmiererfahrene Administratoren angebunden werden können. Insbesondere sollten die Administratoren auch über entsprechende Sicherheitskenntnisse verfügen (bspw. WS-Security).

Alle Systeme bieten Paymentfunktionen an. Joomla!, Drupal und Plone verwenden dazu eigene Extensions.

Zur leichteren und sicheren Integration in Systemumgebungen ist die Verwendung von Standards zur Authentisierung hilfreich. Besonders die Verfahren OpenID und OAuth erlauben eine Site-übergreifende Authentisierung, wobei die Verwendung dieser Verfahren auch eine sorgfältige Konfiguration zur Vermeidung von größeren Sicherheitslücken erfordert. Grundsätzlich werden diese Verfahren von allen

88 http://www.oasis-ws-i.org/

Systemen angeboten. Besonders flexibel sind dabei TYPO3, WordPress und Plone, die drei oder mehr Verfahren gleichzeitig unterstützen. TYPO3 unterstützt bspw. OpenID und OAuth, RADIUS, IMAP sowie LDAP über Extensions. WordPress bietet auch OpenID, OAuth sowie LDAP. Joomla! unterstützt dagegen nur OpenID und LDAP – weitere Protokolle sind aber über Extensions realisierbar. Drupal bietet OAuth und LDAP sowie OpenID je über eine Extension an. Plone unterstützt neben OpenID, OAuth, LDAP auch über die Extension „WebServerAuth“ Kerberos.

4.2.3.4 Sichere Architektur: Skalierbarkeit des Systems

Alle betrachteten Systeme verfügen über eigene Caching-Mechanismen, um die CPU-Last bei der Auslieferung von Inhalten gering zu halten. Dabei verwenden alle Systeme mit Ausnahme von Plone ein CMS spezifisches Caching. Plone benötigt die Erweiterung „plone.app.caching“, die eine Steuerung des Caching Verhaltens ermöglicht. Das ausgewählte Verfahren wie bspw. „Strong caching“ oder „Weak caching“ kann dann von Caching-Proxies wie Varnish oder Squid umgesetzt werden. Weitere Details für alle CMS hierzu finden sich in den Tabellen im Anhang unter dem Kriterium „Nutzung von Caching Mechanismen“ ab Seite 99.

4.2.3.5 Sichere Architektur: Auswahl und Reife der Erweiterungsmechanismen

Die Auswahl und Reife der Erweiterungsmechanismen und hier insbesondere die saubere Trennung der APIs nach Erweiterungsaspekten, konnte im Rahmen dieser Studie nicht vertieft geprüft werden. Grundsätzlich bieten aber alle Systeme Erweiterungslösungen an.

Die Verwendung von Erweiterungen ist natürlich nicht immer ganz risikofrei, da man hiermit bei schlecht programmierten Erweiterungen Sicherheitslücken und im ungünstigsten Fall auch vorsätzlich eingebrachter Schadcode, der unter Umständen weitgehende Rechte auf dem Server besitzt, auf dem System installiert. Deshalb ist unbedingt darauf zu achten, dass die verwendeten Erweiterungen aus vertrauenswürdigen Quellen stammen und man im Zweifel sich genau über die gewünschte Erweiterung, bspw. auf den entsprechenden Foren des Systems, informiert.

Erweiterungen von TYPO3 werden im Backend mit Hilfe des Extension-Managers administriert. Dieser listet alle möglichen Erweiterungen auf. Mittels ein- und ausklicken werden diese aktiviert bzw. deaktiviert. Die Extensions werden über die TYPO3 Extension API eingebunden, wobei drei Haupttypen, nämlich Plugins, Module und Services, verfügbar sind. Ein Plugin kann bspw. ein Shop System sein. Ein Modul ist eine Funktionserweiterung für das Backend und Services stellen neue Funktionen über eine eigene API sowohl für das Frontend als auch für das Backend zur Verfügung.

Die WordPress Plugin API kann Erweiterungen über zwei Typen von „Hooks“ einbinden. „Actions“ werden durch bestimmte Ereignisse bei der Action API ausgelöst wie bspw. die Veröffentlichung eines Textes und „Filter“ kontrollieren den Datenfluss zwischen Datenbank und Browser, also die Generierung der Seiten durch WordPress bzw. umgekehrt die Übernahme von Daten in WordPress.

Joomla! unterscheidet zwischen Modulen und Plugins. Module sind Erweiterungen, die das Rendern der Seiten beeinflussen können und besitzen i.d.R. eine geringere Komplexität als Plugins. Joomla! Plugins unterscheiden sich in Content, Search, Authentication und System Plugins. Alle Plugin-Klassen sind sogenannte „observer classes“, die an ein globales Event-Dispatcher Objekt im Joomla!-Kern gebunden sind. Damit ist es möglich, Joomla! ereignisgesteuert mit neuen Funktionen zu erweitern.

Auch Drupal bietet die Möglichkeit, Erweiterungen (bei Drupal „Modules“ genannt) zu entwickeln. Hierzu stehen eine Reihe von sogenannten „Drupal Hooks“, d.h. Funktionen des Drupal Cores zur Verfügung, woran die Module, abhängig von ihrer Funktionalität, angeknüpft werden.

Plone verwendet zur Erweiterung „add-ons“, die als Python Packages entwickelt und zur Verfügung gestellt werden. Diese werden dann auch über PyPi (Python Package Index)[89] verbreitet. Zur Reduktion der Komplexität von Python Packages werden Plone ZopeSkel Code Templates (Code Skeleton Templates) bereitgestellt, auf deren Basis bspw. neue Themen oder Funktionalitäten entwickelt werden können.

4.2.3.6 Umsetzung von Sicherheitsanforderungen: Rollen und Rechtekonzept

Die Vererbung von Rechten im Content-Modell wird von WordPress und Joomla! realisiert.

Alle Systeme haben ein Rollen- oder Gruppen-basiertes Rechtesystem, mit dem sich abgestufte Zugriffsrechte für eine größere Anzahl von Benutzern definieren lassen. Eine attributspezifische Rechtevergabe (ABAC) ist aber bei keinem der Systeme implementiert. Bei den Systemen TYPO3, Joomla!, Drupal und Plone lassen sich die Inhalte einer Website hinsichtlich unterschiedlicher Zugriffsrechte sehr detailreich gliedern.

4.2.3.7 Umsetzung von Sicherheitsanforderungen: Anpassbarkeit

Alle CMS verfügen über Möglichkeiten zur individuellen Anpassung. Beispiele dafür sind die Verwendung eines speziellen URL-Formats oder die Umsetzung einer Version der Web-Seite für mobile Endgeräte. In der Regel werden solche Anpassungen über externe Erweiterungen realisiert. Bei TYPO3 ist der ist Zugriff mobiler Geräte auch für das Back-End geplant.

Nicenames, die die Wartbarkeit, Navigation und Übersichtlichkeit erhöhen, werden von allen Systemen angeboten.

Mit Ausnahme von Joomla! und Drupal bieten auch alle Systeme Funktionen zur Personalisierung an, die eine individuelle Gestaltung der Webseiten unterstützen.

Die Kontextsensivität ist nur bei Drupal und Plone dokumentiert. Ob die übrigen Systeme ähnliche Funktionen besitzen, die nur noch nicht ausreichend dokumentiert sind oder ob sie tatsächlich keine derartige Funktion besitzen, konnte nicht weiter überprüft werden.

4.2.3.8 Umsetzung von Sicherheitsanforderungen: Sprachvarianten

Die durchgehende Unterstützung von Unicode (UTF-8) ist nicht nur zur Unterstützung verschiedener Sprachen hilfreich, sondern dient auch der Vermeidung von Sicherheitslücken durch falsch zugeordnete Encodings. Erfreulicherweise ist Unicode-Unterstützung bei allen Systemen gegeben.

4.2.3.9 Umsetzung von Sicherheitsanforderungen: Sichere Datenablage

Grundsätzlich ist es für CMS wichtig, dass sie eine sichere Datenablage realisieren, so dass die Inhalte auch tatsächlich nur von den jeweils berechtigten Zielgruppen gelesen bzw. auch verändert werden können.

Die Ablage von Zugangsdaten wird von den untersuchten CMS mit sehr unterschiedlicher Qualität umgesetzt.

TYPO3 speichert das Benutzer-Passwort mittels der Extension „salted passwords“ als MD5-Hash, mittels Portable PHP Password Hashing Framework (phpass) oder Blowfish in jeweils einer Tabelle, wobei Back-End- und Front-End-User in getrennten Tabellen abgelegt werden. MD5 entspricht dabei nicht mehr den aktuellen Mindestanforderungen der Bundesnetzagentur für die Verwendung von Hash-Algorithmen und sollte daher vermieden werden.

89 https://pypi.python.org/pypi?%3Aaction=search&term=plone&submit=search

WordPress nutzt auch phpass, das Portable PHP password hashing framework. Passwörter werden mit den Algorithmen blowfish, extended DES oder MD5 (mit Salt und mehreren Iterationen) gehashed und in einer Datenbank-Tabelle abgelegt. Welcher Algorithmus verwendet wird, hängt davon ab, was jeweils bei der verwendeten Installation zur Verfügung steht – MD5 wird nur als letzte Option verwendet, sofern auf dem System keine anderen Algorithmen verfügbar sind.

Joomla! benutzt für das Benutzerpasswort den MD5-Hash mit Salt in einer Tabelle. Dies entspricht nicht mehr den aktuellen Mindestanforderungen der Bundesnetzagentur für die Verwendung von Hash-Algorithmen.

Drupal speichert das Benutzer-Passwort auch als Hashwert in einer Tabelle. Der Algorithmus basiert auf SHA512 mit Salt mit mehreren Hash-Iterationen. Die Zahl der Iterationen liegt zwischen 128 (27) und etwa einer Milliarde (230) – sie soll mit jeder Drupal-Version verdoppelt werden, um dem Leistungszuwachs von zum Cracking verwendeten Computern entgegenzuwirken. Momentan (Drupal Version 7) werden 32768 Hash-Iterationen (215) durchgeführt.

Plone verwendet SSHA, d.h. Salt und SHA. Dieser Algorithmus kann mit LDAP, Kerberos und der Zope-Datenbank (ZODB) verwendet werden. Zope verwaltet User und bietet Hashing der Passwörter „out of the box“. Plugins erhalten maximal die Hashwerte der Passwörter von den normalen User-Objekten, so dass diese die User-Objekte nicht weiter auswerten können.

Alle Systeme bieten die Möglichkeit der Rücksicherung der Daten im Ganzen (bspw. über sqldump), jedoch nur WordPress, Drupal und Plone unterstützen auch die Rücksicherung einzelner Bereiche des Datenbestandes. Bei Plone wird dies über Scripte realisiert.

Besonders vertrauliche Nutzerdaten sollten in besonderen Bereichen mit höherem Schutzniveau gespeichert werden können. Alle Systeme bieten natürlich die Möglichkeit der Transportverschlüsselung über SSL. Weitergehende Verfahren wie bspw. die individuelle Verschlüsselung von Inhaltsdaten wird leider von keinem System standardmäßig angeboten. Drupal bietet aber als einfache Alternative zur Verschlüsselung per SSL das Modul „Encrypt Submissions“, das eine Verschlüsselung von Web-Formulardaten erlaubt. Diese werden durch das Plugin jCryption, das AES mit 256-bit nutzt, verschlüsselt bzw. auf Serverseite mittels PHP entschlüsselt. Jedoch muss für die Nutzung Java Script auf dem Client aktiviert sein, ansonsten erfolgt eine unverschlüsselte Übertragung. Es handelt sich aber auch hier nur um eine reine Transportverschlüsselung.

Die Trennung von fachlichen Daten und Systemdaten ist für den Datenschutz besonders wichtig, da dann der technische Betrieb nicht in die fachlichen Daten, die ggf. vertraulich sind, einsehen muss. Leider ist auch diese Funktionalität bei keinem der untersuchten CMS standardmäßig verfügbar.

4.2.3.10 Umsetzung von Sicherheitsanforderungen: Suchmaschine

Alle Systeme bieten detaillierte Suchfunktionen. Die Verwendung von Standardsuchmaschinen hat gegenüber eigenen, proprietären Lösungen den Vorteil der vereinfachten Wartbarkeit, insbesondere im Falle von auftretenden Sicherheitslücken. Eine Standardsuchmaschine steht für fast alle Systeme zur Verfügung. TYPO3 verwendet Apache Solr über eine Extension, WordPress verwendet Solr mittels Plugin (standardmäßig wird eine eigene Lösung verwendet), Drupal, Joomla! und Plone bieten auch Solr als Suchmaschine an.

4.2.3.11 Umsetzung von Sicherheitsanforderungen: Inhaltsverwaltung

Bei der Erstellung und Verwaltung von Inhalten durch Redakteure bewegen sich alle Systeme auf hohem Niveau: Unterstützt werden redaktionelle Workflows wie mehrstufige Freigabeprozesse, die Versionierung und Wiederverwendung von Inhalten, sowie die Erstellung von Voransichten (Previews). Die Systeme lassen sich somit gut an Arbeitsabläufe anpassen, wodurch unsichere Praktiken und Notbehelfe wie die Weitergabe von Passwörtern oder Arbeiten mit Administratorrechten vermieden werden können.

4.2.3.12 Umsetzung von Sicherheitsanforderungen: Authentifizierung

Sichere Verfahren zur Authentifizierung sind essentiell für ein Content Management System.

Fast alle Systeme bieten eine Registrierung durch den Benutzer selbst an, lediglich bei TYPO3 ist die Registrierung nicht Bestandteil der Standardinstallation. Nur Joomla! nutzt eine 2-pass-registration, bei der der Benutzeraccount erst nach Aktivierung eines per Mail zugesandten Links endgültig angelegt wird. Die anderen Systeme können evtl. durch Massenregistrierung überlastet werden. Eine generelle Schwachstelle der Benutzer-Selbstregistrierung ist, dass nach Eingabe einer Mailadresse offenbar wird, ob ein anderer Nutzer mit dieser Adresse bereits eingetragen ist. Mit diesem Wissen können Passwort-Reset-Mails (s.u.) ausgelöst, evtl. abgefangen und missbraucht werden.

Bei keinem System außer WordPress wurden Schwachstellen beim Anmeldevorgang entdeckt. WordPress hingegen verwendet unterschiedliche Meldungen bei Login-Fehlversuchen mit existierendem oder unbekanntem Nutzernamen (vgl. Abbildungen 4.1 und 4.2). Damit kann ein Angreifer zunächst einen korrekten Loginnamen feststellen und anschließend Anmeldeversuche gezielt auf diesen Account konzentrieren.

Abbildung 4.1: Existierender Benutzername, falsches Passwort

Bei Drupal ist ein Schutz vor Brute Force Angriffen bereits im Grundsystem integriert. Bei den anderen Systemen kann er über CMS-Erweiterungen oder andere Produkte wie das Apache-Modul mod_security realisiert werden. Alle diese Lösungen verfahren so, dass nach einer festgelegten Zahl von fehlerhaften Anmeldungen innerhalb einer vorgegebenen Zeit weitere Versuche oder Zugriffe dieses Benutzers oder dieser IP-Adresse blockiert werden. Dieser Mechanismus kann allerdings auch für Denial-of-Service-Attacken missbraucht werden, indem ein Benutzer durch mehrere fehlerhafte Anmeldevorgänge in seinem Namen ausgesperrt wird; die Dauer der Sperre sollte deshalb mit Bedacht gewählt werden. Eine Sperre von 5 Minuten nach mehr als 5 Loginfehlern würde einem automatisierten Brute-Force-Angriff das Ausprobieren von nur noch 72 Passwörtern pro Stunde ermöglichen, für den Benutzer ist die Wartezeit aber akzeptabel. Bei länger eingestellten Sperrdauern ist ein Eingreifen durch den Administrator erforderlich; ein manuelles Rücknehmen der Sperre ist jedoch bei Drupal und dem Joomla!-Plugin Brute Force Stop nicht vorgesehen.

Abbildung 4.2: Nicht existierender Benutzername, beliebiges Passwort

Einen Mechanismus für Benutzer, die ihr Passwort vergessen haben, bieten alle Systeme an. Schwachstellen wurden nicht entdeckt.

Positiv hervorzuheben ist das von Drupal verwendete Formular zum erstmaligen Setzen oder zum Ändern eines Passworts: Um die Nutzer zur Verwendung sicherer Passwörter zu motivieren, wird während der Passworteingabe ein „Stärke-Balken“ angezeigt und es werden auf fünf Eigenschaften Hinweise gegeben, die die Sicherheit des Passworts erhöhen. Mit jedem eingegebenen Zeichen werden diese beiden Anzeigen angepasst (vgl. Abbildung 4.3bis 4.5).

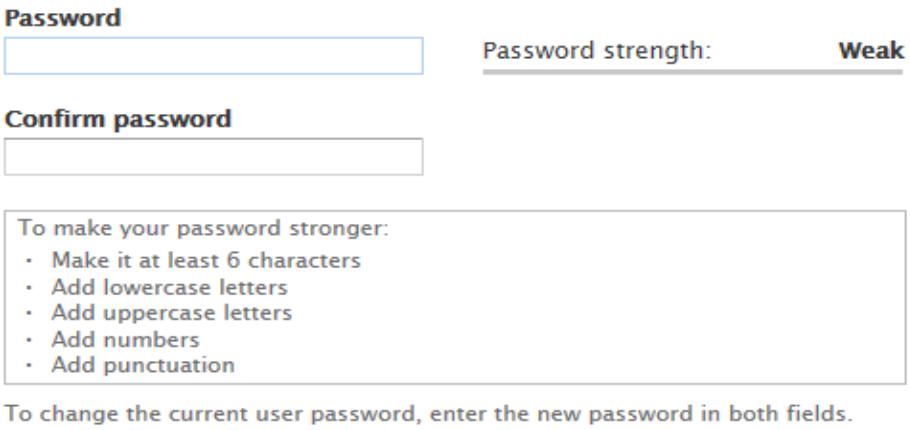


Abbildung 4.3: Zu Beginn bei leerer Eingabe

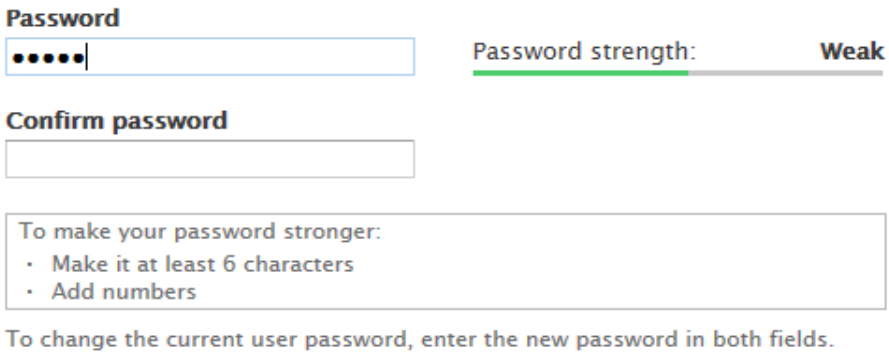

Password
Password strength:
Weak
Confirm password
To make your password stronger:
Make it at least 6 characters
Add numbers
To change the current user password, enter the new password in both fields.

Abbildung 4.4: Nach Eingabe von PW%pa

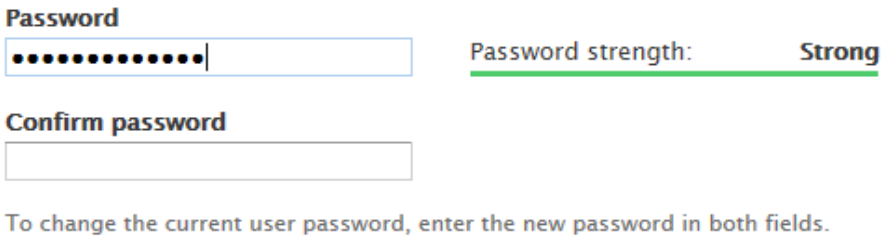


Abbildung 4.5: Nach Eingabe von PW%passwort+1

Captcha Funktionalitäten sind mittels Plugins für alle Systeme verfügbar (vgl. Tabellen im Anhang ab Seite 99).

4.2.3.13 Umsetzung von Sicherheitsanforderungen: Sitzungsverwaltung

Auch eine sichere Sitzungsverwaltung ist für CMS essentiell. Hier ist insbesondere ein CSRF-Schutz wichtig sowie die Verwendung von Secure- und HttpOnly-Flags bei Cookies.

Alle Systeme unterstützten die entsprechenden Maßnahmen für eine sichere Sitzungsverwaltung.

4.3 Service Transition

4.3.1 Installation (Deployment)

4.3.1.1 Vorbedingungen/Anforderungen

Selbstverständlich sollten CMS nicht mit Root-Rechten auf dem Server betrieben werden. Aber auch die Installation sollte möglichst ohne Root-Rechte erfolgen, so dass es möglich ist, das CMS bei vorkonfiguriertem LAMP-Stack bspw. auf einem gemieteten Server eines Providers, der natürlich seinen Kunden i.d.R. keine Root-Rechte überlassen möchte, zu installieren. TYPO3, WordPress und Joomla! werden mit den Rechten von Apache (bzw. des IIS unter Windows) installiert und betrieben und benötigen für den Betrieb keine Root-Rechte. Drupal und Plone werden bei der Installation mit einem eigenen User ohne Root-Rechte installiert. Dieser muss ggf. vorkonfiguriert sein. Die Systeme werden dann mit den jeweiligen Benutzerrechten des eigenen Users betrieben.

Als PHP-basierte Systeme erfordern und benutzen TYPO3, WordPress, Joomla! und Drupal keinen Paketmanager oder ein Build-System. Für Plone ist mit buildout ein Skript- und Recipe-gesteuertes Buildsystem verfügbar.

Die Systeme TYPO3, WordPress, Joomla! und Drupal bestehen aus PHP-Skripten und liefern bei der Installation keine weiteren Bibliotheken mit. Sie verwenden die auf dem Server vorher zu installierenden Komponenten wie Webserver, Datenbank und PHP, für deren Installation und Aktualisierung der Serverbetreiber verantwortlich ist. Plone wird für die Standardinstallation mit dem „Unified Installer“ ausgeliefert, der alle benötigten Komponenten wie Python und das Zope 2 Framework enthält. Es werden auch hier keine zusätzlichen Bibliotheken installiert.

4.3.1.2 Komplexität

Die Anzahl der manuellen Arbeitsschritte bei Installation und Konfiguration ist bei den betrachteten Systemen unterschiedlich:

- Für TYPO3 müssen eine Datenbank angelegt und die entsprechenden Zugangsdaten in eine Konfigurationsdatei eingetragen werden. Weitere Einstellungen werden durch ein Installationsskript vorgenommen. Anschließend muss aber der Zugriff auf sicherheitsrelevante Dateien und Verzeichnisse eingeschränkt werden, ein Account und Passwörter müssen gelöscht oder geändert werden.
- Auch für WordPress muss nur die Datenbankanbindung in einer Konfigurationsdatei eingetragen werden.
- Bei Joomla! sind alle Konfigurationsarbeiten Teil eines geführten Installationsdialogs.
- Auch für Drupal muss zunächst die Datenbank eingerichtet werden. Der Zugriff auf die Konfigurationsdatei muss ermöglicht werden, damit das anschließend aufzurufende Installationsskript sie ergänzen kann. Schließlich muss die Konfigurationsdatei wieder zum Schreiben gesperrt werden. Außerdem sollte der durch die Installation angelegte user#1 entfernt werden. Schließlich sind noch Cron-Jobs u.a. zur regelmäßigen Wartung des Drupal-Systems zu erzeugen.
- Bei Plone sind keine zusätzlichen Schritte zur Installation notwendig.

Ein automatisiertes Update des CMS steht für WordPress und Joomla! zur Verfügung; für TYPO3 ist es für die Version 6.1 vorgesehen. In Drupal müssen das System und die eingesetzten Module manuell aktualisiert werden, was aber durch entsprechende Cron-Jobs zumindest teilweise automatisiert werden kann. Zur Plone-Aktualisierung stehen Updateskripte bereit, deren Einsatz mittels buildout zumindest teilweise vereinfacht werden kann.

Grundsätzlich gilt bei allen Systemen, dass eine sichere Default-Konfiguration die Risiken einer unsicheren Konfiguration für den Betrieb minimiert. Werden bspw. Standardpasswörter für die Administratorenkonten verwendet, müssen diese unverzüglich nach der Installation geändert werden, denn bei Nichtänderung stellen diese natürlich ein erhebliches Risiko dar. Die TYPO3-Defaultinstallation ist in einigen Punkten unsicher, bspw. Secure Cookies sind deaktiviert oder beim Backend-Zugang wird https nicht erzwungen. WordPress enthält in der Standardkonfiguration keine separat aktivierbare Absicherung des Backend-Zugangs mit https. Auch Joomla! erlaubt nach der Installation einen Backend-Zugang ohne Transportverschlüsselung über https. „Force SSL“ muss nachträglich für den Administrator aktiviert werden. Bei Drupal sollte der User #1 wieder entfernt werden, Plone generiert für ein benutzerdefiniertes Konto mit Manager-Rechten ein zufälliges Passwort.

Hinweise in den Konfigurationsdateien zur Bedeutung der jeweiligen Parameter gibt es bei TYPO3 kaum. Die Konfigurationsdatei von WordPress hingegen ist ausführlich dokumentiert und kommentiert. Die mit Joomla! gelieferte Musterdatei hatte einige wichtige Hinweise, die daraus vom Setup erzeugte Datei allerdings keinen einzigen mehr. Für Drupal und Plone gibt es keine zentrale Konfigurationsdatei.

Bei Updates werden die sicherheitsrelevanten Einstellungen der Konfigurationsdateien durch TYPO3, WordPress und Joomla! problemlos übernommen, bei Drupal muss z.B. die evtl. wichtige Datei .htaccess manuell wiederhergestellt werden, für Plone konnte dies nicht geprüft werden.

Bei allen Systemen ist eine Installation ohne oder mit nur wenigen zusätzlichen Arbeitsschritten möglich. WordPress, Joomla! und ab der nächsten Version TYPO3 können auch automatisiert aktualisiert werden. Dabei werden i.d.R. die sicherheitsrelevanten Einstellungen (außer bei Drupal) übernommen, soweit wir das prüfen konnten.

4.3.2 Anwenderdokumentation

4.3.2.1 Lieferung Guidelines / Tutorials

Zu allen betrachteten Systemen sind Tutorials oder Guidelines verfügbar, bei denen aber nicht immer Sicherheit das Thema ist. TYPO3 bietet u.a. einen ausführlichen Security Guide, weiteres Schulungsmaterial und einen Blog des Security Teams. Bei WordPress findet sich kein Dokument, das sich ausdrücklich mit sicherheitsrelevanten Aspekten befasst. Für Joomla! gibt es neben rund zehn Artikeln der Serie Security

Checklist etwa dreißig Artikel zu einzelnen sicherheitsrelevanten Themen. Zu Drupal gibt es einen Administration & Security Guide. Plone bietet einen gut strukturierten Überblick über die Sicherheit. Weitere Details hierzu finden sich in den Tabellen im Anhang (vgl. Seite 99).

4.3.2.2 Qualität: Zielgruppenorientierung

Zielgruppengerechte Aufteilung der Dokumentation findet sich bei allen Systemen, bei Joomla! sogar durch fünf Reader Profiles auf der Einstiegsseite der Dokumentation und noch weitere acht Rollen.

4.3.2.3 Qualität: Navigation und Suche

Navigation und Suche in der Dokumentation sind bei den Systemen TYPO3 und WordPress durch den Aufbau als Wiki implizit realisiert. Die Dokumentation zu Joomla! ist gut strukturiert, besonders die zielgruppenspezifischen Quick Links sind hilfreich. Die Einstiegsseite zur Drupal-Dokumentation ist gut strukturiert, die einzelnen Themenseiten enthalten die erwarteten Informationen und die Suchfunktion findet die relevanten Dokumente. Die Navigation im Plone User Manual und den anderen verfügbaren Handbüchern ist gut gelöst, die Suche nach einem Stichwort dagegen findet selbst die entsprechende gleichnamige Manual-Seite nicht.

4.3.2.4 Qualität: Mehrsprachigkeit

Bei keinem System liegt die gesamte Dokumentation in deutsch und englisch vor. Die Dokumentation zu TYPO3 und WordPress liegt nur auf englisch vor. Das Projekt Joomla! Security hat begonnen, deutschsprachige Dokumente zum Thema Sicherheit in Joomla! bereitzustellen. Unter drupalcenter.de erarbeitet eine Community derzeit ein deutschsprachiges Handbuch zu Drupal. Bei www.plone.de findet man manche Dokumente in deutsch, darunter aber nichts Sicherheitsrelevantes.

4.3.2.5 Qualität: Aktualität

Für die meisten Systeme ist die zugehörige Dokumentation größtenteils auf einem aktuellen Stand, mit Ausnahme von TYPO3, wo es einige Mängel gibt. Man kann sich aber bei TYPO3 über die Security Bulletins auf dem neuesten Stand zur Sicherheitsthematik halten. Bei Joomla! sind noch nicht alle Dokumente für die aktuelle Version 3.0 erhältlich, aber bis auf einzelne Dokumente sind die zur Version 2.5 gehörenden weiter anwendbar.

4.3.2.6 Qualität: Vollständigkeit

Eine Beschreibung der Sicherheitsmechanismen findet sich bei fast allen Systemen, wenn auch nicht immer in dem benötigten Umfang:

- Der Security Guide von TYPO3 beschreibt typische Risiken und Bedrohungen und die verfügbaren Mittel, eine Site davor zu schützen.
- Zu WordPress enthält das Dokument Hardening WordPress entsprechende Informationen.
- Die Guidelines und FAQ zu Joomla! enthalten zwar Beschreibungen einiger Mechanismen, aber keine tiefergehenden Informationen.
- Bei Drupal liefert der Abschnitt „Securing your Site“ im Administration & Security Guide diese Informationen.
- Der Artikel „Hardening Plone“ in der Plone Knowledge Base beschreibt zwar entsprechende Use Cases, aber nicht alle Sicherheitsmechanismen.

Auch eine Beschreibung der sicherheitsrelevanten Einstellungen bieten fast alle Systeme. Lediglich für Joomla! haben wir keine Übersicht der Einstellungen gefunden.

Eine Beschreibung von Szenarien für verschiedene CMS-Einsatzzwecke wie extensives Caching, Replikation Single Sign On haben wir nur bei zwei Systemen ansatzweise gefunden:

- Bei WordPress wird der Einsatz für Multiple Blogs, Gruppen-Blogs oder als CMS beschrieben.
- Die bei Plone beschrieben Szenarien betreffen Single Sign On mit Active Directory, eine robuste Plone/Zope-Installation und die Installation unter verschiedenen Betriebssystemen.

Kein System bietet eine im Sinne dieser Kriterien vollständige Dokumentation an.

4.4 Service Operation

4.4.1 Marktdurchdringung

4.4.1.1 Internetpräsenz

Zu allen untersuchten Content Management Systemen gibt es Foren, in denen zum Teil auch Sicherheitsaspekte diskutiert werden. Aber spezielle Foren zu Sicherheitsthemen finden sich nur für Joomla! und Drupal. Bei Joomla! gibt es je eines für jede der drei Versionen.

Fast alle Entwicklerteams haben eine spezielle Gruppe von Mitarbeitern, die sich mit Sicherheitsfragen befassen, die Security Teams oder Security Strike Teams.

4.4.1.2 Anzahl der Installationen

Die Anzahl der Installationen in Deutschland der jeweiligen CMS ist nicht exakt feststellbar und beruht auf unterschiedlichen Quellen und Messverfahren. Sie liegt für alle betrachteten Systeme aber deutlich über 50 Installationen.

Alle betrachteten CMS haben eine ausreichende Installationsbasis und widmen einen Teil der Ressourcen den Sicherheitsaspekten; für Joomla! und Drupal können die Anwender auf spezielle Sicherheitsforen zurückgreifen.

4.4.2 Behandlung von Änderungen/Security Patches

4.4.2.1 Security Patches

Die Zeit bis zum Erscheinen eines Security Patch, gerechnet von der Meldung einer Schwachstelle, konnten wir nicht ermitteln, da diese Daten i.d.R. nicht veröffentlicht werden. Einige Aussagen von Mitgliedern aus

Security Teams geben an, dass eine Behebung von Lücken innerhalb einer Woche angestrebt wird, einzelne Stichproben in den Foren belegen aber, dass es manchmal auch länger dauern kann. Weitere Details und Links hierzu finden sich in den Tabellen im Anhang ab Seite 99 unter dem Kriterium „Behandlung von Änderungen“.

Die Begleitinformationen zu Security Patches sind unterschiedlich detailliert:

- Die Releases von TYPO3 haben jeweils einen Changelog, eine Beschreibung von Änderungen und Verbesserungen sowie Hinweise zur Kompatibilität mit früheren Versionen.
- Für die Versionen von WordPress sind eine Liste der geänderten Dateien, ein detaillierter Changelog, eine Zusammenfassung der Änderungen sowie eine Übersicht der durch die Version behobenen Fehlermeldungen verfügbar. Allerdings sind die Informationen nicht immer vollständig: Eine in der Zusammenfassung erwähnte XSS-Schwachstelle wird in der Liste der behobenen Fehler nicht genannt.
- Joomla! liefert im Release-Announcement eine Liste der behobenen Fehler und der neuen Features, zusammen mit der Angabe der betroffenen Releases und einer Upgrade-Empfehlung. Die Detailinformationen zu Security Issues enthalten die CVE Number und eine allgemeine Beschreibung der Sicherheitslücke, die Angaben zu anderen Issues geben eine detaillierte Beschreibung des Problems und der vorgenommenen Änderungen.
- Die Information zu einem Drupal Patch, ein Security Advisory, beschreibt die durch den Patch behobenen Lücken einschließlich des zugehörigen CVE-Identifiers, nennt die betroffenen Drupal-Versionen und gibt, falls nötig, weitere Hinweise zur Sicherheit an.
- Plone nennt in seinen Security Advisories die betroffenen Plone-Versionen, nennt die behobenen Schwachstellen und stellt unter einem separaten Link auch die notwendigen Patches und Hotfixes zur Verfügung.

Von den CMS, für die diese Kriterien geprüft werden konnten, liefert lediglich Plone eher dürftige Informationen zu den Security Patches.

4.4.3 Betrieb

4.4.3.1 Logging: Revisionsfähigkeit

Der für eine Revisionsfähigkeit erforderliche Schutz der Integrität der Logeinträge durch Zeitstempel und Signatur ist bei allen Systemen nicht gegeben. Der Schutz der Integrität der Logeinträge kann aber meist durch den Einsatz eines Logservers erreicht werden.

Die ebenfalls für Revisionsfähigkeit erforderliche Vollständigkeit der Logeinträge (Urheber und Zeitpunkt der vermerkten Aktionen) wird von Joomla! und Drupal angeboten, von Plone bei Verwendung des Publication Control Systems. Bei TYPO3 kann ein Logeintrag zwar einem Nutzeraccount aber keiner Session (Zeitangabe) zugeordnet werden. Für WordPress sind nur technische Logs verfügbar und für Joomla! und Drupal war nicht feststellbar, ob und wie fachliche Logs konfiguriert werden können.

4.4.3.2 Logging: LogLevel-konfigurierbar

Unterschiedliche Loglevel sind bei TYPO3, WordPress, Drupal und Plone einstellbar; bei Joomla! kann nur das Logging für Fehler beeinflusst werden.

Komplett revisionssicheres Logging ist bei keinem der Systeme mit Bordmitteln möglich, aber meist durch den Einsatz eines Logservers erreichbar. Sieht man vom Schutz der Integrität ab, so bieten Joomla!, Drupal und Plone zumindest vollständige Logeinträge mit einstellbarem Detaillierungsgrad.

4.4.3.3 Logging: Vertraulichkeit

In TYPO3 können die Logeinträge zwar über einen Filter in der Anzeige getrennt werden, durch Deaktivieren des Filters können technische Administratoren aber auch die fachlichen Logeinträge einsehen. Von wem die in Plone erzeugbaren fachlichen Logs eingesehen werden können, konnte nicht geprüft werden. In WordPress werden nur technische (PHP-Error-)Logs erzeugt, für Joomla! und Drupal konnte dieses Kriterium nicht geprüft werden.

Eine Verschlüsselung der fachlichen Logs ist in TYPO3 nicht möglich, da sie zusammen mit den technischen Logs in einer gemeinsamen Datenbanktabelle abgelegt werden. Für Drupal gibt es eine Erweiterung, mit der Datenbankeinträge verschlüsselt werden können; falls fachliche Logs möglich sind (s.o.) und in der Datenbank abgelegt werden, erfüllt Drupal dieses Kriterium. WordPress unterstützt keine fachlichen Logs, für Joomla! und Plone konnte das Kriterium nicht geprüft werden.

Ob eine Rücksicherung von Informationen aus der Archivierung Klartext offenbaren würde, konnte im Rahmen dieser Studie nicht geprüft werden.

4.4.3.4 Einbindung in Systemmanagement: Laufzeitinformationen extrahierbar

In den Dokumentationen der untersuchten CMS wurden keine Hinweise zur Abfrage von Laufzeitinformationen durch ein Systemmanagement-Werkzeug gefunden.

Einbindung in Systemmanagement: Steuerung per Skript

TYPO3 erlaubt eine Steuerung per Skript und stellt hierfür eine eigene Skriptsprache, TypoScript, zur Verfügung. Auch die Systeme Drupal (bietet eine Skriptsprache auf der Basis von Lisp) und Plone können über Skripte in ihren Grundfunktionen (Start, Stop, Neuladen der Konfiguration) gesteuert werden. Für WordPress und Joomla! standen keine Informationen zur Einbindung in ein Systemmanagement per Skript zur Verfügung.

Einbindung in Systemmanagement: Steuerung per API

Eine Steuerung der oben genannten Grundfunktionen über API ist bei TYPO3, WordPress und Drupal möglich. Für Joomla! und Plone standen keine Informationen zur Einbindung in ein Systemmanagement per API zur Verfügung.

4.5 Zusammenfassung der Sicherheitsuntersuchung

Alle Systeme bieten die Möglichkeit, das CMS vor einer Entscheidung auszuprobieren und sich über den Funktionsumfang zu informieren.

Fehler, insbesondere sicherheitsrelevante, können gemeldet werden und werden meist in angemessener Zeit behoben. Alle Systeme informieren den Nutzer auch durch Advisories zu sicherheitsrelevanten Informationen, nachdem die Fehler behoben wurden.

Die Hersteller haben Maßnahmen für eine sichere Entwicklung wie Qualitätskriterien, Richtlinien, Reviews oder Tests vorgesehen, wobei über einen messbaren Erfolg der Maßnahmen nichts konkret gesagt werden kann. Die Anzahl und Verteilung der gefundenen Schwachstellen lässt aber indirekt Aussagen zu (vgl. Kapitel 4.6).

Sicherheitsrelevante Aspekte der Architektur wie Modularisierung, Trennung von Systemkomponenten, Verwendung von Standards werden bei allen Systemen berücksichtigt.

Die Verwendung von Diensten wie Web Services, Payment, Authentisierung wird von den Systemen unterschiedlich realisiert. Hier ist bei einer Entscheidung darauf zu achten, ob eigene oder Standardverfahren genutzt werden. Web Services stehen nur für Betreiber/Administratoren mit Programmier- und Sicher-

heitserfahrung in diesem Umfeld zur Verfügung. Joomla!, Drupal und Plone verwenden für Paymentfunktionen eigene Verfahren. Hervorzuheben sind TYPO3, WordPress und Plone, die mehrere Standardverfahren zur Authentisierung unterstützen.

Caching-Mechanismen zur Erhöhung der Performance bieten alle Systeme, eine Session-Replikation auf mehrere Server nur Plone. Die PHP Systeme können einen Opcode-Cache verwenden, der zusätzliche Laufzeitgewinne bringt.

Alle CMS haben ein Rollen- oder Gruppen-basiertes Rechtesystem, wobei die Systeme unterschiedliche Verfeinerungsmöglichkeiten bieten (vgl. Kapitel 4.2.3.6).

Eine individuelle Anpassung beispielsweise mittels spezieller URL-Formate, Nicenames, Personalisierung ist bei allen Systemen vorgesehen. Auch UTF-8 steht bei allen CMS zur Verfügung.

Eine sichere Ablage von Zugangsdaten ist nicht überall gegeben. Die Trennung der Datenablage nach Schutzbedarf sowie eine Teilbereiche betreffende Rücksicherung ist nirgendwo gegeben (vgl. Kapitel 4.2.3.9).

Alle Systeme bieten Mechanismen zur Inhalteverwaltung (Workflows, Freigabe, Versionierung etc.).

Sichere Verfahren zur Authentifizierung (Brute-Force-Schutz, Captchas, vergessene Passwörter) und zur Sitzungsverwaltung (CSRF-Schutz) gibt es für alle Systeme, teilweise aber nur als Erweiterung. Für diese essentiellen Eigenschaften und deren Stärke sind die Tabellen im Anhang zu beachten.

Die Erstinstallation der Systeme ist stark unterschiedlich, die Skala reicht von einer komfortablen strengen Skript-Führung mit fast keinen Nacharbeiten (Plone) bis zu umfangreichen Konfigurationsmöglichkeiten, die den Administrator in ihrer Menge zu erschlagen drohen (TYPO3). Regelmäßige Updates können bei Drupal, WordPress, Joomla! automatisiert werden, bei TYPO3 steht das auf dem Releaseplan.

Die Aufteilung der Anwenderdokumentation ist bei allen Systemen zielgruppengerecht. Die Aktualität der Dokumentation ist bei TYPO3 oft zu verbessern. Bei einigen CMS sind sicherheitsrelevante Aspekte in der Dokumentation allerdings nicht ausreichend behandelt (vgl. Kapitel 4.2.1.5).

Eine Beschreibung der Sicherheitsmechanismen und -einstellungen sowie der Realisierung von Szenarien findet sich zusammenhängend nur bei Plone, alle übrigen CMS bieten nur Einzeldokumente zu den verschiedenen Mechanismen (vgl. Kapitel 4.2.1.6).

Ein revisionssicheres Logging lässt sich bei keinem der Systeme mit Bordmitteln realisieren. Ebenso läßt sich die Vertraulichkeit von fachlichen Logs nur über Betriebssystemmittel hinzufügen.

Die Voraussetzungen für ein effizientes Einbinden des CMS in eine Systemmanagement-Umgebung ist bei keinem CMS vollständig gegeben. Logging und Tracing lässt sich bei allen Systemen gut konfigurieren, die Skriptfähigkeit von Drupal und Plone ist positiv hervorzuheben.

4.6 Abgleich mit der aktuellen Bedrohungslage

In diesem Abschnitt soll ein Versuch unternommen werden, Zusammenhänge zwischen der Bewertung anhand der Untersuchungskriterien und den Ergebnissen der Bedrohungsanalyse in Kapitel 3 zu ermitteln.

Die Anzahl der gemeldeten Schwachstellen (vgl. Abbildung 3.9) – im Mittel etwa 14 pro Jahr und CMS – kann als Beleg dafür gewertet werden, dass die von allen Open-Source-Systemen vorgesehenen Meldewege (vgl. Kapitel 4.2.1.3) angemessen sind und genutzt werden.

Dafür, dass ein Großteil der Schwachstellen in den Erweiterungen der CMS zu finden ist (vgl. Tabelle 3.23 und Abbildung 3.8), gibt es mehrere mögliche Erklärungen:

- Zu einem über einen längeren Zeitraum relativ stabilen Kernsystem gibt es eine Vielzahl von Erweiterungen, die evtl. von unterschiedlichen Teams entwickelt und überarbeitet werden. (mit Ausnahme von TYPO3, wo drei unterschiedliche Kerne von unterschiedlichen Teams gepflegt werden).

- Die Security Teams der Hersteller befassen sich in der Regel mit Schwachstellen in den Kernkomponenten der Systeme. Viele der verfügbaren Erweiterungen fallen nicht in ihren Bereich; auf die Bereinigung der Fehler in diesen Erweiterungen haben sie keinen oder nur wenig Einfluss. So ist auch zu erklären, dass Schwachstellen in den Kernkomponenten der Systeme häufiger beseitigt werden als solche in Erweiterungen (vgl. Tabelle 3.24).
- Zu allen Systemen sind Qualitätskriterien und Richtlinien vorhanden (vgl. Kapitel 4.2.1.4). Organisatorische Maßnahmen um die Einhaltung dieser Kriterien und Richtlinien auch durchzusetzen sind aber bei Joomla! nicht dokumentiert und greifen bei TYPO3 für Erweiterungen nur unter bestimmten Bedingungen. Allgemein können die CMS-Entwickler die Qualität und Sicherheit von Erweiterungen nicht sicherstellen, die außerhalb der Projektplattformen angeboten werden.

Erweiterungen für das Python-basierte CMS Plone werden üblicherweise als Python Package ausgehend von einem durch die Plone-Entwickler vorgegebenen Code Skeleton Template entwickelt. Durch diese vorgegebene Struktur und die Verwendung von Zope-Komponenten werden einige Quellen für Schwachstellen vermieden. So lässt sich die im Vergleich zu den anderen CMS geringe Zahl von gemeldeten Plone-Schwachstellen erklären.

Zusammenfassend kann gesagt werden, dass ausreichende Richtlinien und Qualitätskriterien zusammen mit entsprechenden Reviews und Sicherheitstests helfen, das Risiko von Schwachstellen in den Basissystemen zu reduzieren, bei benötigten Erweiterungen aber im Einzelfall darauf geachtet werden soll, ob sie ebenfalls Tests unterzogen und von einem Security Team betreut werden.

5 Zusammenfassung der Studie

5.1 Ergebnisse

5.1.1 Sicherheitsniveau der betrachteten CMS

Zuerst darf festgestellt werden, dass die betrachteten Open Source Projekte nachweislich einen Sicherheitsprozess implementiert haben. Die Software hat Produktcharakter mit einem veröffentlichten Releaseplan, einem transparenten Bugtracker etc. Die Umsetzung eines Sicherheitsprozesses entspricht dem Stand der Technik, den selbst viele unter Zeitdruck erstellte kommerzielle Softwarepakete nicht erreichen. Die resultierende Software ist – gemessen an ihrer Funktionalität und der daraus resultierenden Komplexität – eine gute Wahl für einen Dienstanbieter.

Der konfigurative Aufwand beim Installieren eines CMS ist ungleich höher als bei einer spezialisierten Webapplikation. Dies ergibt sich aus den vielen verschiedenen Anwendungsfällen und Rahmenbedingungen, in denen das CMS lauffähig sein soll. Diese Konfiguration beeinflusst maßgeblich die effektive Sicherheit der Website.

Keines der betrachteten Systeme kann jedoch „as is“, unbeobachtet oder durch den unbedarften Anwender betrieben werden. Mehr noch als spezialisierte Webapplikationen unterliegen Content Management Systeme dem permanenten Wettstreit zwischen Exploit-Entwicklern und Security Teams. Anzahl und Frequenz der veröffentlichten Schwachstellen erfordern bei den betrachteten CMS ein geschätztes Zeitbudget von täglich ca. 15 min pro Website, um verfügbare Patches zu erkennen, Datensicherungen vorzunehmen und Patches einzupflegen. Diese Schätzung geht davon aus, dass tägliche Datensicherungen bereits Teil des Systemmanagements sind. Sobald ein Dienstanbieter diesen Aufwand trägt, erhält er den Nutzen einer, im Vergleich zu einer spezialisierten Webapplikation, deutlich höheren Testabdeckung, d.h. eines deutlich geringeren Risikos unentdeckter Schwachstellen.

5.1.2 Optimierungsmöglichkeiten

Der sichere Betrieb einer CMS-Website resultiert aus drei hauptsächlichen Faktoren:

- (1) der Sicherheit der verwendeten Software,
- (2) der angemessenen Konfiguration und
- (3) dem angemessenen Systemmanagement (einschließlich des Patchmanagements).

Der Fokus der Projekte liegt bisher auf (1) und (3). Das Schließen der Lücken, die durch unsichere Konfigurationen entstehen, wird einen deutlichen Gewinn an Sicherheit bringen. Wie oben dargestellt, lagern viele CMS architektonische Aufgaben wie Lastverteilung, Hochverfügbarkeit etc. an andere Komponenten aus. Auch sollte das Sicherheitssystem immer „mehrschichtig“ sein, d.h. sich nicht auf die Wirksamkeit einer einzelnen Schicht verlassen. Deshalb muss die „sichere Konfiguration“ übergreifend über die Infrastruktur auf Basis von Hauptanwendungszwecken unterstützt werden. Dies ist nicht allein Aufgabe des CMS Projektes. Wo die CMS dies jedoch unterstützen könnten, wäre bei der Standardisierung von Konfigurationschecks (vgl. Kapitel 5.3.2).

EMPFEHLUNG Eine „sichere Konfiguration“ muss übergreifend über die Infrastruktur auf Basis von Hauptanwendungszwecken unterstützt werden.

Um (1) besser prüfen zu können, wäre es hilfreich, wenn die Open Source Projekte die periodisch durchgeführten Schwachstellentests standardisieren und veröffentlichen würden. Und um über (1) bessere Gewissheit zu haben, wäre es förderlich, wenn die Open Source Projekte regelmäßig automatisch statische Quellcode Analysen insbesondere über die Erweiterungsmodule durchführen und die offenen Punkte als Info veröffentlichen würden. So wären die Entscheidungen über die Verwendung von Erweiterungsmodulen noch leichter zu treffen.

Die Struktur und der Informationsgehalt bei der Veröffentlichung von Schwachstellen sind bei Drupal, TYPO3, Joomla! (feed) und Plone musterhaft, während sie bei WordPress noch verbessert werden können.

Keines der Systeme stellt Webservice-Schnittstellen bereit, die nach WS-Security abgesichert sind. Dies mag an mangelndem Bedarf oder an zu großem Aufwand liegen. Es wäre wünschenswert, im Sinne einer „defense in depth“ die Kommunikation zwischen CMS und SOA-Diensten Ende-zu-Ende absichern zu können. WS-Security ist im Bereich Java- oder .Net ein gut unterstützter Standard. Des Weiteren wäre im Sinne einer „defense in depth“ zu begrüßen, wenn die exponierten CMS ihre Credentials zum Zugriff auf Datenbanken nicht lokal abspeichern, sondern (ggf. periodisch) von einem Dienst beziehen würden. Dann könnte man über das Systemmanagement die Anfrage nach neuen Credentials mit anderen Ereignissen im System, z.B. dem Hochfahren des CMS korrelieren. Dann wäre es möglich, dass nur unter bestimmten Bedingungen ein neues Credential angefordert werden kann. Die Hürde für einen Angreifer, der bereits den Server übernommen hat, würde höher werden.

Im Bereich der Autorisierung setzt sich XACML als Standard immer mehr durch. Der Ansatz ist es, infrastrukturübergreifend Zugriffsrechte zu spezifizieren, welche durch die einzelnen Komponenten umgesetzt werden. Es wäre wünschenswert, auch Content Management Systeme an eine solche zentrale Autorisierung anknüpfen zu können.

Die Trennung der Informationsablagen nach Schutzbedarf ist ein wünschenswertes Feature. Es ist ein immer wiederkehrendes Problem, dass Administratoren mit „root“-Rechten alle fachlichen Daten sehen können. Die Lösung dieses Problems besteht darin, die Komponenten nur noch in Ausnahmefällen mit Auditierung als „root“ zu administrieren. Dies funktioniert jedoch nur, wenn die Software das Management mit einem Administratorenkonto ohne „root“-Rechte ermöglicht. Die Trennung in unterschiedlichen Schutzbedarf ermöglicht es auch sehr gut, technische Logs von fachlichen zu trennen.

Die Dokumentation sicherheitsrelevanter Einstellungen an Ort und Stelle ist bei Plone und WordPress gut und bei TYPO3 verbesserungswürdig.

5.2 Szenario-basierte Handlungsempfehlungen

Alle betrachteten CMS bieten dem Anwender eine Fülle von Funktionen, Erweiterungen und Ausbaustufen. Grundlage für die hier getroffenen Aussagen sind Installationen der Szenarien im Labor, d.h. ausgehend von den bisherigen Untersuchungsergebnissen dokumentiert dieses Kapitel Auswahlüberlegungen, die sich auf die Anwendungsszenarien im Kapitel 2.4 beziehen. Hierbei werden auch einige für das jeweilige Szenario zu erwartende Bedrohungen sowie Sicherheitsmaßnahmen bzw. Funktionen der CMS betrachtet.
Grundsätzlich gilt, dass alle Systeme mit mehr oder weniger Zusatzaufwand für die unten stehenden Szenarien nutzbar sind.

5.2.1 Szenario 1: „Private Event Site“

Die öffentliche, zeitweise verfügbare Website ist das einfachste beschriebene Szenario, welches die untere Grenze an Anforderungen darstellt, für die noch ein CMS als technische Lösung in Betracht kommt. Dennoch darf die Privatsphäre der beteiligten Personen nicht außer Acht gelassen werden. Funktionen zur Sicherstellung von Vertraulichkeit und Integrität sind notwendig, gerade weil sie für den (unbedarften) Anwender eine untergeordnete Rolle spielen und weil nicht davon auszugehen ist, dass Mühe und Zeit in die Absicherung der Website investiert werden. Ein Minimum an Maßnahmen ist notwendig, damit Bedrohungen wie beispielsweise:

- die Platzierung von Schadfunktionen auf dem Server zur Beobachtung/Überwachung der Besucher oder zur Verbreitung von Schadsoftware,
- das Hochladen von unangemessenen Inhalten durch unberechtigte Dritte,
- der Missbrauch des Servers für die Nutzung in Botnetzen,
- verschiedene Formen des Identitätsdiebstahls

weitgehend vermieden werden. Es wird angenommen, dass die Redakteure der Website so umsichtig sind, keine privaten Daten abzulegen. Deshalb dürfen wir von einem normalen Schutzbedarf[90] ausgehen. Dies erfordert dennoch, dass die Sicherheitsfunktionalitäten des CMS mit minimalem Zutun des Anwenders aktiviert sein müssen. Da die Website für den Eigentümer keine wirklich große Bedeutung besitzt, muss auch davon ausgegangen werden, dass sie nach dem Event eine gewisse Zeit „unbeaufsichtigt“ weiter existiert. Sicherheitsmaßnahmen bzw. CMS-Funktionalitäten für dieses Szenario sind beispielsweise:

- die einfache und sichere Handhabung von Passwörtern der Redakteure (z.B. durch Hinweise auf die Passwortsicherheit) inklusive der sicheren Wiederherstellung von vergessenen Passwörtern,
- Maßnahmen zur Vermeidung von automatisierten Aktivitäten auf der Website (CAPTCHA[91]),
- eine einfache und sichere Administration, die keine allgemein bekannten Standard-Accounts und -Passwörter zulässt und
- ein weitgehend automatisiertes Update-Management, so dass insbesondere Sicherheitspatches ohne Zutun des Dienstanbieters eingespielt werden.

Für dieses Szenario lautet die Empfehlung, keine eigene Website aufzubauen, sondern einen Dienst aus dem inzwischen breiten Spektrum professioneller Dienstleister[92] in Anspruch zu nehmen. Diese Dienstleister sorgen im eigenen Interesse für professionelles Management und schalten die Website ggf. auch ab, falls sie missbraucht wird. Die Basissoftware ist hier uninteressant, grundsätzlich können alle betrachteten Content Management Systeme so vorbereitet werden, dass der im Szenario angesprochene Dienstanbieter für die beschriebenen Zwecke damit umgehen kann. Das Problem bei einer eigenen Website ist die Komplexität der Einrichtung – nicht nur des CMS, sondern auch des Webservers und der Datenbank – und die Erforderlichkeit permanenter „Beaufsichtigung“.

5.2.2 Szenario 2: „Bürgerbüro in einer kleinen Gemeinde“

Das im Kapitel 2.4.2 beschriebene Szenario konzentriert sich insbesondere auf Merkmale wie:

- eine gut funktionierende Suche,
- die Möglichkeit, ein Kontaktformular zu benutzen,

90 Weiterführende Informationen hierzu unter: https://www.bsi.bund.de/DE/Themen/weitereThemen/WebkursITGrundschutz/Schutzbedarfsfeststellung/Schutzbedarfskategorien/Abgrenzung/abgrenzung_node.html

91 **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part

92 vgl. http://en.wikipedia.org/wiki/Comparison_of_free_web_hosting_services

- eine ordentliche Kalenderfunktion für die Präsentation von wichtigen Terminen und
- die Anbindung an einen LDAP-Server für die Benutzerverwaltung.

Sicherheit spielt in diesem Szenario eine besondere Rolle, da personenbezogene Daten verarbeitet werden. Vertraulichkeit und Integrität haben hier einen deutlich höheren Stellenwert. Auch die Anforderung an die Verfügbarkeit ist hoch zu bewerten, da die Kommune mit dem elektronischen Angebot ihre Kosten verringern möchte. Die Bürger müssen darauf vertrauen können, dass dieser Dienst ständig erreichbar ist und funktioniert. Für die Anwendung wird deshalb von hohem und ggf. sehr hohem Schutzbedarf für die personenbezogenen Daten ausgegangen. Für diesen Server muss von der kleinen Gemeinde mindestens der BSI-Standard 100-2 IT-Grundschutzvorgehensweise[93] beachtet werden, was aber in diesem Kontext nicht weiter vertieft werden soll, jedoch durchaus einen zu beachtenden Kostenfaktor darstellt.

Beispiele für zu erwartende Bedrohungen sind:

- die Platzierung von Schadfunktionen auf dem Server zur Beobachtung/Überwachung der Besucher oder zur Verbreitung von Schadsoftware, das Hochladen von unangemessenen und strafrechtlich relevanten Inhalten durch unberechtigte Dritte,
- der Missbrauch des Servers für die Nutzung in Botnetzen,
- das Abhören der Kommunikation mit dem Server,
- Angriffsszenarien zur Einschränkung der Verfügbarkeit sowie
- das Ausspionieren und Verbreiten der personenbezogenen Daten und der Accounts sowie alle Formen des Identitätsdiebstahls.

Es muss bei diesem Szenario davon ausgegangen werden, dass die personenbezogenen Daten auf keinen Fall unverschlüsselt in der Datenbank des CMS abgelegt werden dürfen. Die Daten sollten in eine besonders gesicherte Datenbank eines Fremdsystems überführt werden, die dem Anspruch des hohen bzw. sehr hohen Schutzbedarfs genügt. Die damit verbundene Vorgehensweise und entsprechende Maßnahmen für das Fremdsystem werden an dieser Stelle nicht weiter betrachtet, jedoch muss beachtet werden, dass die ggf. am Front-End des CMS eingegebenen oder auch angezeigten personenbezogenen Daten dem Schutzbedarf entsprechend abgesichert werden.

Sicherheitsmaßnahmen bzw. Funktionalitäten des CMS für dieses Szenario sind dann beispielsweise:

- die einfache und sichere Handhabung von Passwörtern (z.B. durch Hinweise auf die Passwortsicherheit) inklusive der sicheren Registrierung, der Wiederherstellung von vergessenen Passwörtern,
- Maßnahmen zur Vermeidung von automatisierten Aktivitäten auf der Website (CAPTCHA),
- eine einfache und sichere Administration, die keine allgemein bekannten Standard-Accounts und -Passwörter zulässt,
- ein weitgehend automatisiertes Update-Management, so dass insbesondere Sicherheitspatches automatisiert durch den dafür vorgesehenen Mitarbeiter der Kommune eingespielt werden. Das Szenario beschreibt, dass es genau eine Person gibt, die die Pflege der Website einschließlich des Patchmanagements übernimmt,
- die Möglichkeit, vorgefertigte Erweiterungen (Kalender, Kontaktformular) zu installieren und ggf. anzupassen,
- der Betrieb sowohl des Front- als auch des Back-Ends mittels HTTPS-Protokolls zur Sicherstellung der notwendigen Transportverschlüsselung,

93 Weiterführende Informationen hierzu unter: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html#doc471418bodyText2

- eine sichere Anbindung der verwendeten Datenbank, die nicht auf dem gleichen Server betrieben wird und deren Inhalte separat geschützt, d.h. mit sicheren Mechanismen/Parametern verschlüsselt sind,
- die Möglichkeit, einen externen Verzeichnisserver zur Nutzerverwaltung sicher anzubinden.

Der Schutzbedarf „hoch“ impliziert, dass die Infrastruktur aus Firewall, Webserver, CMS, Datenbank, LDAP und anderen erforderlichen Diensten sorgfältig aufeinander abgestimmt und gehärtet wird.

Für dieses Szenario sind Drupal und Plone gut geeignet. Für Drupal ist, wie bereits erwähnt, die Lernkurve für das Erstellen eigener Web-Seiten zu Beginn sehr steil. Insofern sollte der damit beauftragte Administrator eine vorgefertigte Drupal-Distribution nutzen. Beide Systeme Plone und Drupal zeichnen sich durch einen sehr gereiften Sicherheitsprozess aus. Ein grundlegendes Prinzip der Plone/Python Community lautete „no surprise“. Unter Berücksichtigung der im Szenario dargestellten sehr limitierten personellen Ressourcen ist dies das führende Prinzip.

Sofern die Prozesse im Bürgerbüro über organisatorische Maßnahmen so geändert werden könnten, dass über die Website keine personenbezogenen Daten erfasst und verarbeitet werden, gestaltet sich die Absicherung der Website entschieden einfacher. Bei normalem Schutzbedarf, insbesondere auch dann, wenn die laut Szenario mit Pflege und Betrieb betraute Person nur über geringere IT-Kenntnisse verfügt, sind auch Joomla! und WordPress eine gute Wahl. Jedoch sollte dann auf die Verwaltung von personenbezogenen Daten vollständig verzichtet werden.

5.2.3 Szenario 3: „Open Government Site einer Kleinstadt“

Das Szenario konzentriert sich zusätzlich zu den schon im Szenario 2 betrachteten Inhalten auf weitere Merkmale wie:

- die Einrichtung einer Community,
- die Möglichkeit, Umfragen zu erstellen, Newsletter zu publizieren, User Generated Content zu präsentieren und
- die Möglichkeit einer Anbindung an Geschäftsprozesse über Web Services zu realisieren.

Außerdem wären eine rollenbasierte Nutzerverwaltung, eine bessere Suchmaschine sowie Mehrsprachigkeit von Vorteil. Sicherheit spielt natürlich auch in diesem Szenario eine besondere Rolle, da hier bei der Verwendung von personenbezogenen Daten bspw. bei Mechanismen zur Bürgerbeteiligung und der Anbindung von Geschäftsprozessen sichergestellt werden muss, dass diese entsprechend der Anforderungen des Datenschutzes realisiert werden. Vertraulichkeit, Verfügbarkeit, Integrität und ggf. Authentizität sowie Verbindlichkeit, hergestellt durch elektronische Signaturverfahren, haben hier einen besonders hohen Stellenwert. Es muss für die Funktionen zur Bürgerbeteiligung und die Geschäftsprozesse von hohem und ggf. sehr hohem Schutzbedarf insbesondere bei der Nutzung personenbezogener Daten ausgegangen werden. Für diesen Server muss von der Kleinstadt der BSI-Standard 100-2 IT-Grundschutzvorgehensweise[94] beachtet werden und darüber hinaus muss der BSI-Standard 100-3: Risikoanalyse[95] auf der Basis von IT-Grundschutz sowie der BSI-Standard 100-4: Notfallmanagement[96] in Betracht gezogen werden. Die Umsetzung dieser Standards soll aber in diesem Kontext nicht weiter vertieft werden. Beispiele für zu erwartende Bedrohungen unterscheiden sich kaum von Szenario 2. Auch hier muss von Bedrohungen wie:

94 Weiterführende Informationen hierzu unter: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html#doc471418bodyText2

95 Weiterführende Informationen hierzu unter: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html#doc471418bodyText3

96 Weiterführende Informationen hierzu unter: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html#doc471418bodyText4

- der Platzierung von Schadfunktionen auf dem Server zur Beobachtung/Überwachung der Besucher oder zur Verbreitung von Schadsoftware,
- dem Hochladen von unangemessenen und strafrechtlich relevanten Inhalten durch unberechtigte Dritte,
- dem Missbrauch des Servers für die Nutzung in Botnetzen,
- dem Abhören der Kommunikation mit dem Server,
- Angriffsszenarien zur Einschränkung der Verfügbarkeit sowie
- dem Ausspionieren und Verbreiten der personenbezogenen Daten der Geschäftsprozesse und der Accounts sowie allen Formen des Identitätsdiebstahls

ausgegangen werden. Auch hier sollten die personenbezogenen Daten und weitere Geschäftsprozessdaten nicht unverschlüsselt in der Datenbank des CMS abgelegt werden. Die Daten benötigen eine strukturell gesicherte Datenbank des jeweiligen angebundenen Geschäftsprozesses, die dem Anspruch des hohen bzw. sehr hohen Schutzbedarfs genügt. Alle Infrastrukturkomponenten von der Lastverteilung über Webserver, CMS, Datenbank bis zur Servicelandschaft der Hintergrundsysteme sollten konzeptionell gegeneinander geschützt sein und ihre Kommunikationsverbindungen angemessen abgesichert sein (Verteidigung in der Tiefe). Die damit verbundene Vorgehensweise und entsprechende Maßnahmen für die Geschäftsprozesse werden an dieser Stelle nicht weiter betrachtet, jedoch müssen die am CMS eingegebenen oder auch angezeigten Daten dem Schutzbedarf des jeweiligen Geschäftsprozesses entsprechend abgesichert werden.

Sicherheitsmaßnahmen bzw. Funktionalitäten des CMS für dieses Szenario sind beispielsweise:

- die einfache und sichere Handhabung von Passwörtern (z.B. durch Hinweise auf die Passwortsicherheit) inklusive der sicheren Registrierung und der Wiederherstellung von vergessenen Passwörtern,
- Maßnahmen zur Vermeidung von automatisierten Aktivitäten auf der Website (CAPTCHA[97]),
- eine einfache und sichere Administration, die keine allgemein bekannten Standard-Accounts und -Passwörter zulässt,
- ein gut definiertes Update-Management, so dass insbesondere Sicherheitspatches zeitnah eingespielt werden,
- die Möglichkeit, vorgefertigte Erweiterungen für Community-Websites (Umfragen, Bewertungen, Newsletter, User Generated Content) zu installieren und ggf. anzupassen,
- der Betrieb sowohl des Front- als auch des Backends mittels HTTPS-Protokolls zur Sicherstellung der notwendigen Transportverschlüsselung,
- eine sichere Anbindung der verwendeten Datenbank, die nicht auf dem gleichen Server betrieben und separat geschützt (verschlüsselt) wird,
- die sichere Anbindung der Geschäftsprozesse über Web-Services, bspw. durch WS-Security, sowie
- die Möglichkeit, einen Verzeichnisserver zur sicheren Nutzerverwaltung anzubinden (auch hier muss ggf. eine Transportverschlüsselung möglich sein).

Das Szenario impliziert, dass bereits ein kleines Team an Konzeption und Betrieb der Website zur Umsetzung der Open Government Geschäftsprozesse mitarbeitet. Die Absicherung der Website für den Schutzbedarf „hoch“ umfasst, wie oben beschreiben, die Härtung der gesamten Infrastruktur. Zusätzlich müssen hier, unabhängig vom CMS, ggf. Zertifikate für eine starke Authentisierung von Benutzern gepflegt und die darauf aufbauende Autorisierung integriert werden. Grundsätzlich müssen alle zusätzlich zu installierenden Erweiterungen auf ihre Reife hin geprüft und ihre Einbindung ggf. in einer Risikobetrachtung abgewogen werden.

97 **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part

Wie im Szenario 2 erscheint Drupal als ein geeignetes System, da es eine Fülle an vorgefertigten Erweiterungen für dieses Szenario mitbringt und sicherheitstechnisch gereift ist. Aber auch Joomla! und TYPO3 bringen für Community- und Social Networking Prozesse eine Vielzahl an Erweiterungen mit, die nicht selbst entwickelt werden müssen. In diesem Falle, unter Berücksichtigung des hohen Schutzbedarfs wäre es eine gute Option, das Sicherheitsmanagement der Joomla! bzw. TYPO3 Infrastruktur komplett auszulagern. Auch Plone mit seiner vergleichsweise geringeren Anzahl von bekannten Sicherheitslücken erscheint als passend. Für Plone ist die Basis vorhandener Community Add-Ons schwächer, hier muss funktional bewertet werden, welche Features tatsächlich benötigt werden und wie viel Aufwand diese Ergänzungen kosten. Keines der betrachteten Systeme bietet jedoch eine Ende-zu-Ende gesicherte Anbindung an die im Szenario beschriebene SOA Landschaft. Dies muss architektonisch anders gelöst werden. Eine Weiterentwicklung der bestehenden Webservice-Erweiterungen um WS-Security wäre eine begrüßenswerte Alternative.

5.2.4 Szenario 4: „Mittelständisches Unternehmen mit mehreren Standorten“

Im letzten Szenario verfügt der Dienstanbieter über ausreichend IT-Personal, um selbst ein professionelles Systemmanagement durchzuführen. Das Szenario, dass im Kapitel 2.4.4 beschrieben wurde, konzentriert sich insbesondere auf Merkmale wie:

- Mandantenfähigkeit für den Betrieb mehrerer Sites,
- einen ausgefeilten Redaktionsworkflow, der an die Prozesse des Unternehmens anpassbar sein muss,
- die Möglichkeit, einen Web-Shop sowie die dafür notwendigen Erweiterungen wie Payment zu betreiben,
- die punktuelle skalierbare Architektur für große Benutzerzahlen sowie
- die Anbindung von Diensten einer SOA.

Vertraulichkeit und Integrität ist auch in diesem Szenario sehr wichtig, da Kundendaten verarbeitet werden und insbesondere durch die Verbindung zu dem vermutlich attraktiven Web-Shop einschließlich des Bezahlsystems mit Angriffen zu rechnen ist. Über die Website werden geschäftskritische Prozesse angestoßen bzw. abgewickelt, die bis in die zentralen Systeme der Firma reichen.

Zusätzlich zu den im Szenario 3 aufgeführten Bedrohungen muss damit gerechnet werden, dass

- professionelle Angreifer über die Website in das interne Firmennetz einzudringen versuchen und
- Angriffe auf die Shoplösung und die Verarbeitung der Transaktionsdaten (Kunden-Details, Kreditkartendaten etc.) stattfinden.

Anfallende personenbezogene Daten aus dem Shopsystem und auch weitere Geschäftsprozessdaten dürfen deshalb auf keinen Fall unverschlüsselt in den Datenbanken des CMS abgelegt werden. Das Shopsystem muss die entsprechenden Standards (SET) implementieren, so dass die Kreditkartendaten gar nicht auf der Site verarbeitet werden.

Zusätzliche Sicherheitsmaßnahmen bzw. Funktionalitäten der CMS für dieses Szenario sind:

- die leichte und sichere Integration der Shop- und Bezahlsysteme, einschließlich der dafür genutzten Datenbank,
- die sichere Anbindung der Geschäftsprozesse, sowie
- die Möglichkeit, einen externen Verzeichnisserver zur sicheren Nutzerverwaltung anzubinden (auch hier muss ggf. eine Transportverschlüsselung möglich sein).

Die in diesem Szenario geforderte Einbindung von Fremdsystemen per SOA kann von keinem der untersuchten CMS direkt realisiert werden. Hier müssen im Einzelfall Lösungen eingekauft oder selbst entwickelt werden. Für den Workflow eines mittelständischen Unternehmens erscheinen TYPO3 und Drupal, die u.a.

ein Shopsystem und auch einen konfigurierbaren Workflow anbieten, als geeignet. Joomla! und WordPress bieten kaum geringere Möglichkeiten. Hier ist ein professionelles Sicherheitsmanagement aufzusetzen. Plone bietet kaum Alternativen bei den Shopsystemen. Im konkreten Einzelfall muss das Unternehmen selbst prüfen, welches System sich mit dem geringsten Aufwand in die Unternehmensprozesse einbinden lässt und die geeigneten Funktionalitäten für den zu erstellenden Anforderungskatalog bietet.

5.3 Ausblick

In einem sich rasant entwickelnden Umfeld stellt sich immer die Frage, ob die gestern gewonnenen Erkenntnisse heute noch von Bedeutung sind und was man an allgemein gültigen Anschauungen aus dem Text ziehen kann. Die in Kapitel 3 dargestellten Statistiken wurden im Dezember 2012 erstellt, der Abgleich der Sicherheitsuntersuchung mit diesem Stand fand im März 2013 statt. Da die Datenlage statistisch sehr instabil ist (wenige Daten verfügbar) können neue Releases, neue Schwachstellenfunde, neue Priorisierungen innerhalb der Extensions die Erkenntnisse der Bedrohungslage (Kapitel 3) und der Sicherheitsuntersuchung (Kapitel 4) relativ schnell ändern, während die Aussagen in der Zusammenfassung (Kapitel 5) mittelfristige Gültigkeit besitzen.

Hier werden maßgebliche Tendenzen diskutiert, die eine individuelle Betrachtung bereichern können.

5.3.1 Entwicklung der Gefährdungen (OWASP Top 10 2013)

Die Top 10 sind die durchschnittlich risikoreichsten Schwachstellen. Da Websites mit Content Management Systemen einen Großteil der Websites allgemein ausmachen, gehen die hier auffällig gewordenen Probleme natürlich in die Liste mit ein:

- A5-Security Misconfiguration: Dies ist besonders bei CMS ein gravierendes Problem. Wenn man die Annahme trifft, dass besonders Nicht-Informatiker dafür verantwortlich sind, ergibt sich die Frage, (1) wie sich der Anteil der Nicht-Informatiker unter den Administratoren von CMS-Sites entwickelt.
  Dazu kommen weitere zum Teil gegenläufige Faktoren: (2) Die IT-Produkte versuchen, immer „alltagstauglicher“ zu werden. Es besteht ein lebendiger Wettstreit um „Kunden“. Für die Akzeptanz der potenziellen Anwender ist „leichte Administrierbarkeit“ ein wichtiges Argument. Dadurch entsteht bei einem Prozentsatz der potenziellen Administratoren der Irrglaube, auch den Sicherheitsanforderungen der Website gewachsen zu sein.
  (3) Es besteht ein starker Trend weg vom Informatiker, Administrator „um die Ecke“, den man kennt und als Person schätzt zum Hosting oder Cloud Service, zu den Administratoren im Datacenter des Providers, der tausende Sites gleichzeitig professionell verwaltet.
  (4) die Entwicklung des Sachverstandes und der Security Awareness bei Nicht-Informatikern ist ein weiterer Faktor, der sehr schwer zu beurteilen ist. Die Parallele zu anderen komplexen Systemen zeigt jedoch, dass in einem fortgeschrittenen Entwicklungsstadium auch komplexe Systeme für Nicht-Experten handhabbar sind. Meist gibt es vereinfachte, limitierte Sichten auf das System, die Bedienfehler auf ein akzeptables Minimum senken. Dies geht zu Lasten der bestimmbaren Systemeigenschaften, weshalb für spezielle Einsatzbereiche wieder professionelle Unterstützung eingebunden wird. Deshalb ist in Summe davon auszugehen, dass die Anzahl an fehlkonfigurierten CMS-Sites mittelfristig sinken wird, so dass die Position von „A-5 Security Misconfiguration“ zumindest für CMS-Sites stark fallen wird.
- A9-Using Components with Known Vulnerabilities: Dieser Punkt behandelt die Frage des Patch-Managements. Hier ist bei allen betrachteten CMS die Erkenntnis umgesetzt worden, dass der Benutzer hier zu unterstützen, führen oder nötigen ist. Administratoren werden oft deutlich darauf hingewiesen, dass Patches verfügbar sind. Ähnlich den Beobachtungen zum Punkt Fehlkonfiguration ist das erzwungene Patchen selbst bei Client Systemen ein gelöstes Problem. Allerdings ist es noch nicht verbreitet, Websites selbsttätig automatisch herunterzufahren, sobald Patches vorliegen. Vielleicht sollte dies als Optimierungsmöglichkeit übernommen werden.

Ein wesentlicher Punkt, der in den untersuchten CMS bemängelt und hier explizit als eigene Kategorie aufgeführt wurde:

- A7-Missing Function Level Access Control: Im einfachen Fall wird es einem Angreifer durch fehlende Access Control Mechanismen möglich, sich höhere Rechte zu erschleichen. Verallgemeinert bedeutet es, dass Systeme zu selten mehrschichtige Sicherheitssysteme umsetzen. Systemgrenzen könnten viel mehr dazu genutzt werden, sich sicherheitstechnisch gegeneinander abzugrenzen. Angreifer haben es dann deutlich schwerer, von einem System zum nächsten zu kommen. Dies wird auch mittelfristig noch ein Thema bleiben.

5.3.2 Standards für die sichere Konfiguration

Da die sichere Konfiguration für den sicheren Betrieb derart wichtig ist, und bei jeglichen komplexen Systemen offensichtlich immer fehleranfällig ist, hat das NIST unter Einbindung der Community eine Reihe von Spezifikationen mit dem Ziel erarbeitet, die Konfiguration eines Systems beschreibbar und prüfbar zu machen. Das „Security Content Automation Protocol“ das momentan in Version 1.2 vorliegt enthält das „Extensible Configuration Checklist Description Format“ (XCCDF), die „Open Vulnerability and Assessment Language“ (OVAL®) und die „Open Checklist Interactive Language“ (OCIL). Eine Reihe von bekannten Produkten können darauf basierend automatisiert Konfigurationen testen. Die Spezifikationen sind so aufeinander abgestimmt, dass die vom NIST verwendete Nomenklatura für Schwachstellen und Risiken verwendet werden.

Das zentrale Repository[98] beherbergt die bestehenden Konfigurationsbeschreibungen. Für den Apache Webserver gibt es aktuell für die Version 2.2 fünf verschiedene Konfigurationsprofile zum Download. Keines der betrachteten CMS ist dort vertreten.

Die Idee des Repository trägt dem Umstand Rechnung, dass die Software in verschiedenen Kontexten genutzt wird und sich ständig weiterentwickelt. Die Projekte könnten das Repository mit jedem Release, welches eine neue Konfiguration ermöglicht, selbständig ergänzen. Jede Version könnte mit zugehörigen Metainformationen, z.B. mit einem Schutzniveau, versehen sein. Auch wenn dieser „bausteinorientierte“ Ansatz des NIST ein substanzieller Fortschritt gegenüber allen vergleichbaren Bemühungen ist, wird die Sicherheit von CMS-Sites jedoch häufig durch die umsichtige Verknüpfung vieler verschiedener Komponenten, z.B. Loadbalancer, Webserver, Web application firewall, Caches, Datenbanken erreicht. Es müsste folglich, ähnlich wie in der Autoindustrie üblich, eine Paketierung verschiedener Infrastrukturdienste zu einem „CMS-Cluster“ geben, welches sich an einem typischen Anwendungsszenario (z.B. vgl. Kapitel 5.2) orientiert. Dieses CMS-Cluster könnte als Ganzes beschrieben und damit abprüfbar sein.

5.3.3 Alternative Sicherheitsstrategien

Seit Jahren, fast Jahrzehnten gibt es Forschungen, Strategien und Tools zur Vermeidung von Sicherheitslücken. Das Wissen um mögliche Sicherheitslücken und deren Vermeidung erweitert sich beständig. Gleichzeitig werden immer neue Schwachstellen gefunden und produziert. Die Anzahl der unveröffentlichten Schwachstellen wird vermutlich größer, da ein Markt für funktionierende, noch ungemeldete Exploits (Zero-Day-Exploits) entstanden ist. So ist es nachvollziehbar, dass einige Experten daran zweifeln, dass komplexe Systeme überhaupt abzusichern sind: zu viele Entwickler, zu viele Einflussfaktoren, Anwendungsfälle, eine Wissensbandbreite, die selbst Experten nie in Gänze überschauen.

Aktuelle Strategien[99] gehen vom Gegenteil aus und versuchen komplexere Systeme zu kapseln. Die Intelligenz wird in die Unterscheidung des „normalen“ vom „auffälligen“ Verhalten der Website gesteckt. Sobald das Verhalten auffällig wird, kann der Betreiber prüfen, ob es sich um „reguläre“ gewollte Benutzung

98 http://web.nvd.nist.gov/view/ncp/repository
99 RSA Strategie

der Dienste handelt oder um einen Angriff. So wird entweder eine Sicherheitslücke identifiziert oder die umgebende Kapsel nachjustiert.

Die ENISA Studie „Proactive Detection of Security Incidents"[100] legt großen Dienstanbietern nahe, scheinbar lohnenswerte Angriffsziele als Spielwiese für Angreifer zu erstellen, um sie leicht von den eigentlichen Ressourcen abzulenken und besser aufspüren zu können.

Beide Strategien sind nicht neu, sie sind mit herkömmlichen, ggf. auch Open Source Mitteln, zu realisieren. Beide Ansätze sind valide, sie entbinden den Dienstanbieter jedoch nicht von seinen eigentlichen Pflichten: von einer professionellen Installation, professionellem Systemmanagement und wiederkehrenden externen Sicherheitschecks. Sie können die Sicherheit etwas erhöhen, sind aber kein Ersatz für die täglichen 15 Minuten umsichtigen Handelns des Administrators.

100 http://www.enisa.europa.eu/activities/cert/support/proactive-detection/proactive-detection-of-security-incidents-II-honeypots/at_download/fullReport

Anhang: Faktenübersicht

Nr.	*Fakt*
1	Lastenverteilung wird bei allen PHPH basierte CMS über externe Mittel realisiert.
2	WS-Security wird von keinem der CMS unterstützt.
3	Die Erstellung einer Website mit Drupal erfordert die Einarbeitung in die Basistechnologien, was über rein konfigurative oder administrative Tätigkeiten deutlich hinausgeht.
4	Es gibt bei Drupal prinzipiell ein Recht, ausführbaren Code oder Javascript einzubetten.
5	Drupal ermöglicht über drush die Verwaltung mehrerer Websites auf sichere Art und Weise (SSH-Tunnel).
6	Die Kommunikation der php-basierten Systeme zur mysql Datenbank erfolgt nach dem Stand der Technik.
7	Die Lastverteilung funktioniert bei Plone über den Zope Applikationsserver.
8	Die Komponenten des Plone-CMS kommunizieren über entfernte Prozeduraufrufe, die über SSH getunnelt werden können.
9	Die Administration der Plone Site kann über einen dedizierten Client speziell abgesichert werden.
10	Plone verwendet eine eigene, objektorientierte Datenbank.
11	Die konsequente Reduktion des Systems WordPress auf wesentliche Blogging-Funktionalität ist das herausragende Merkmal gegenüber allen anderen hier betrachteten Systemen.
12	Joomla! verfügt über eine Administrationsoberfläche, die auch für ungeübte Benutzer verständlich und nutzbar ist.
13	Im Unterschied zu WordPress ist Joomla! für Erweiterbarkeit konzipiert, was man an den bereits im Kern vorgesehenen Komponententypen erkennen kann.
14	Die Installation der Core-Features hinsichtlich ihrer Funktionalität ergibt bei Joomla! eine voll nutzbare, per CMS verwaltete Website.
15	Es gibt bei TYPO3 mehrere komfortable Editoren und einen zweistufigen Redaktionsworkflow im Kernsystem.
16	Die traditionelle TYPO3-Linie wird jetzt als CMS 6 weiterentwickelt, während das „next generation CMS“ TYPO3 Neos von Grund auf neu entwickelt wird.
17	Die Anteile der Code-Execution Schwachstellen – die schwerwiegendsten Fehler – sind bei Joomla! und TYPO3 recht hoch.
18	Anhand der Auswertung der Daten zeigt sich klar, dass ein Großteil der Schwachstellen in den Erweiterungen der CMS zu finden sind.

Anhang: Empfehlungsübersicht

Nr.	*Empfehlung*
1	Dienstanbieter müssen permanent in der Lage sein, Patches einzuspielen.
2	Dienstanbieter sollten ihre Websites konzipieren, bevor sie sie aufsetzen. Das Prinzip „Verteidigung in der Tiefe“ ist dabei von herausragender Bedeutung.
3	Dienstanbieter sollten ihre Websites permanent monitoren.
4	Eine „sichere Konfiguration“ muss übergreifend über die Infrastruktur auf Basis von Hauptanwendungszwecken unterstützt werden.

Anhang: Ergebnistabelle TYPO3 CMS 6.0

TYPO3 CMS 6.0				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Design				
Geschäftsführung (Governance)				
Lizenzierung	Ausprobierbarkeit	(+)	http://demo.typo3.org	
Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen		(-)	http://typo3.org/documentation/document-library/core-documentation/doc_core_inside/4.2.0/view/2/10/ http://wiki.typo3.org/Extension_Development,_Security_Guidelines	Dokumentation für Entwickler von Erweiterungen und Security-Abschnitt in Dokumentation für Hauptentwickler scheinen unvollständig und veraltet zu sein.
Transparenz der Kommunikation bei Schwachstellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	(0)	http://typo3.org/teams/security/contact-us/	
	Bug-Tracking-System mit Kategorie Sicherheit	(0)	http://typo3.org/teams/security/security-bulletins/	Die Security Bulletins zeigen Sicherheitslücken und die Lösung dazu.
	Advisories auf der Website des Herstellers leicht auffindbar	(0)	http://typo3.org/teams/security/security-bulletins/	In der Rubrik Support gibt es direkt einen Link zu den Security Bulletins.
	Beschreibung des Prozesses zur Bearbeitung von Sicherheitsproblemen	(0)	http://typo3.org/extension-manuals/doc_guide_security/1.0.1/view/	Vor allem Punkt 1.10 „Detect, Analyze and Repair a Hacked Site“ der Dokumentation beachten.
Sicherer Entwicklungsprozess	Einheitliche Qualitätskriterien	(+)	http://docs.typo3.org/typo3cms/CodingGuidelinesReference/	
	Aussagekräftige und durchgängige Kommentierung im Quelltext	(+)	Quellcode: http://prdownloads.sourceforge.net/typ	

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				o3/typo3_src-4.7.7.zip?download	
	Strukturelle Mittel zur Durchsetzung der Qualitätskriterien		(-)	http://docs.typo3.org/typo3cms/CodingGuidelinesReference/	Eine Beschreibung der formalen Anforderungen bzw. Standards in Bezug auf PHP Programmierung, die beachtet werden sollen, wenn TYPO3 Extensions oder Core Teile entwickelt werden. Es werden keine Verfahren zur Durchsetzung beschrieben.
	Transparenz des Entwicklungsprozesses		(+)	http://typo3.org/news/development/	Der Link führt zu den Development-News von Typo3. Dort wird regelmäßig veröffentlicht, welche Features etc. gerade geplant werden.
Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen			(+)	http://typo3.org/documentation/document-library/core-documentation/doc_core_inside/4.2.0/view/2/10/	
Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene			(-)		Keine vollständige Beschreibung verfügbar.
Prüfung / Test (Verifikation)					
Design-Review nachweislich			(-)		Keine Informationen hierzu verfügbar.
Code-Review nachweislich			(+)	http://typo3.org/teams/security/extension-security-policy/	Unter bestimmten Bedingungen erforderlich für Extensions.
Sicherheitstests nachweislich			(-)	http://docs.typo3.org/typo3cms/SecurityGuide/GeneralGuidelines/StagingServers/Index.html	Mit Hilfe eines Stage Servers sind Sicherheitstest möglich, jedoch nicht in erster Linie dafür vorgesehen.
Softwareentwicklung (Construction)					
Sichere Architektur	Modularisierung / Trennung der	Modularisierung / Trennung der	(0)	http://docs.typo3.org/typo3cms/InsideTypo3Reference/CoreArchitecture/CoreM	Typo3 benutzt das LAMP- bzw. WAMP Prinzip (Linux/Windows, Apache, MySQL, PHP). Die

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	logischen Funktionen	logischen Funktionen		odules/FunctionsModule/Index.html http://www.flagbit.de/typo3-cms/typo3-enterprise-cms/ueber-typo3-cms.html	einzelnen Module können bei Bedarf mit gleichwertiger Software ersetzt werden. Die Funktionen sind entsprechend logisch auf die Module aufgeteilt.
		Trennung in Verschiedene Sicherheitsniveaus möglich (Sandbox)	(0)	http://formidable.typo3.ug/reference/control/sandbox.html	Für Entwicklungszwecke gibt es ein Sandboxing, d.h. das Ausprobieren einer „neuen“, „unsicheren“ Funktion. Dies ist allerdings kein probates Mittel für den Produktivbetrieb, um verschiedene Schutzbedarfe abzubilden.
	Basistechnologie	Verwendung sicherer Bibliotheken	(+)	Testversion	Bei einer Standardinstallation werden keine zusätzlichen Bibliotheken zur verwendeten PhP-Version installiert.
		Ausrichtung an relevanten Standards	(+)	http://wiki.typo3.org/TYPO3_4.7#HTML5_frontend_rendering	HTML5, Accessability (insbes. Government Package)
	Integrationsfähigkeit	Web Services	(0)	http://typo3.org/extension-manuals/typo3_webservice/0.3.8/view/1/1/	Webservices werden über Extension in Typo3 realisiert.
		Payment	(+)	http://typo3.org/extensions/repository/?id=23&L=0&q=payment	Payment Möglichkeiten werden über Extensions in Typo3 realisiert.
		Authentifizierung / Autorisierung	(+)	http://www-ti.informatik.uni-tuebingen.de/~borchert/Troja/studdiplfiles/JonasReichertBachelor.pdf	OpenID OAuth LDAP über Extensions RADIUS IMAP IP-Adressen

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					Die verlinkte Bachelor-Arbeit beinhaltet auf Seite 13 eine Liste der Authentisierungs-mechanismen.
	Skalierbarkeit des Systems	Nutzung von Caching Mechanismen	(+)	http://wiki.typo3.org/Caching_framework	
		Verteiltbarkeit kritischer Komponenten zur Erhöhung Performance/ Verfügbarkeit	(-)	http://wiki.typo3.org/Performance_tuning#TYPO3_performance_Extensions	Die Möglichkeit des Verteilens wird nicht gegeben, jedoch viele Möglichkeiten des Tunings der Komponenten.
	Auswahl und Reife der Erweiterungs-mechanismen		(+)	http://typo3.org/extensions/what-are-extensions/ http://typo3.org/documentation/document-library/core-documentation/doc_core_api/current/view/2/1/	Erweiterungen im Backend werden mit Hilfe des Extension-Managers administriert. Dieser listet alle möglichen Erweiterungen auf und mittels ein-/ausklicken werden diese installiert. Die Trennung der APIs war im Rahmen der Studie nicht prüfbar.
Umsetzung von Sicherheitsanforderungen	Rollen und Rechtekonzept	Vererbung von Rechten	(-)	http://docs.typo3.org/typo3cms/InsideTypo3Reference/CoreArchitecture/AccessControl/Roles/Index.html	Statische Definition von Rollen, keine Vererbung.
		Definition von Gruppen	(+)	http://docs.typo3.org/typo3cms/GettingStartedTutorial/UserManagement/Groups/Index.html	
		Rollen-basierter Zugriff	(+)	http://docs.typo3.org/typo3cms/GettingStartedTutorial/UserManagement/Backe	

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				ndUsers/Index.html	
		Attribut-basierter Zugriff	(-)	http://docs.typo3.org/typo3cms/GettingStartedTutorial/UserManagement/Groups/Index.html	Kein ABAC möglich.
		Granularität der Zugriffssteuerung	(+)	http://docs.typo3.org/typo3cms/GettingStartedTutorial/UserManagement/Groups/Index.html	
	Anpassbarkeit	Unterstützung mobiler Endgeräte	(+)	z.B. http://t3n.de/magazin/einfachen-mittel n-mobilen-website-typo3-iphone-optim ieren-224743/ http://typo3.org/news/article/typo3-60-back-to-the-future/	Nicht out-of-the-box, aber auf verschiedene Weise realisierbar. Für Backend geplant.
		Nicenames / URL-rewriting	(+)	http://wiki.typo3.org/Iis7	Über Extensions realisierbar.
		Personalisierung	(+)	http://www.typo3extensions.org/index.php?title=directmail_personalization	Über Extensions für Mail realisierbar.
		Kontext-Sensitivität	(+)	Franz Ripfel; Irene Höppner; Melanie Meyer „Das TYPO3 Profihandbuch: Der Leitfaden für Entwickler und Administratoren zu Version 4.3“, S. 417 ff	Unter anderem kontextsensitive Hilfen und Menüs
	Sprachvarianten	Internationalisierung nach Best Practices realisiert	(+)	http://wiki.typo3.org/UTF-8_support	

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Sichere Datenablage	Ablage von Zugangsdaten	(+)	Testinstallation, http://typo3extensions.org/index.php?title=t3sec_saltedpw, http://typo3extensions.org/index.php?title=saltedpasswords	Typo3 speichert die Passwörter als salted Hashwerte der Passwörter ab. Die verlinkte Extension ist mittlerweile im Core integriert. Neben dem MD5-Hash – dieser ist laut aktueller Tabelle der Bundesnetzagentur nicht mehr geeignet – kann phpass und blowfish verwendet werden.
		Rücksicherungen konzeptionell vorbereitet?	(-)	http://typo3.org/documentation/document-library/core-documentation/doc_guide_security/1.0.1/view/1/9/	Hinter dem System befindet sich eine MySQL Datenbank. Mit dem Backup-Befehl mysqldump ist es nur möglich ganze Datenbanken und Tabellen zu sichern und auch nur so wiederherzustellen.
		Vertraulichkeit und Integrität von Benutzerdaten bzw. Vorgangsdaten schützbar?	(-)	http://docs.typo3.org/typo3cms/InsideTypo3Reference/CoreArchitecture/Database/Index.html	Daten werden in MySQL Datenbank abgelegt.
		Systemdaten Von fachlichen Daten separiert	(-)		Fachliche Daten sind systemtechnisch nicht separiert.
	Suchmaschine	Reife der Suchmaschine	(+)	http://typo3.org/extension-manuals/doc_indexed_search/current/view/1/1/ http://typo3.org/extension-manuals/solr/current/view/1/1/	Suchmaschinen werden über Extensions integriert, bspw. auch Solr.
	Inhaltsverwaltung	Redaktioneller Workflow	(+)	Testinstallation	Rechtehierarchie erlaubt redaktionellen Workflow.

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		definierbar			
		Versionierung von Inhalten	(+)	http://typo3.org/documentation/document-library/core-documentation/doc_core_inside/4.1.0/view/3/7/	
		Wiederverwendung von Content zur Reduktion der Komplexität	(+)	http://docs.typo3.org/typo3cms/extensions/bs_fce/0.8.0/manual.html http://www.jochenfroehlich.com/en/typo3-best-practice/references.html	Es können Referenzen auf bereits bestehenden Content gemacht werden.
		Preview verfügbar	(+)	http://typo3.org/documentation/document-library/core-documentation/doc_core_inside/4.1.0/view/3/7/	„Previewing: Visiting the frontend website will display it as it will appear when all versions in the workspace is eventually published (switch enable/disable this feature).
	Authentifizierung	Registrierung	(0)	http://typo3.org/extensions/repository/view/sr_feuser_register	Benutzer-Registrierung ist nicht Bestandteil der Standardinstallation, sondern eine Extension. Mailadresse kann für Missbrauch genutzt werden.
		Anmeldung	(0)	Testinstallation	Eine kurze Prüfung auf Sicherheitslücken im Anmeldevorgang hat nichts Auffälliges ergeben. Es waren keine Schwachstellen erkennbar.
		Brute-Force-Schutz (Passwort)	(-)	https://www.dongit.nl/tech/modsecurity-brute-force-protection	Im Typo3-Kern ist kein Schutz gegen Brute-Force-Angriffe gegen Benutzer-Passwörter vorgesehen. Einen rudimentären Schutz verspricht Einsatz der Erweiterung fail2ban oder des Apachemoduls mod_security.
		Passwort vergessen	(-)	Testinstallation	Die Dokumentation hierzu ist mangelhaft. Eine

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				http://www.t3node.com/blog/enhanced-password-recovery-for-frontend-users-in-typo3-43/	weitergehende Prüfung war nicht möglich.
		Captcha	(+)	http://typo3.org/extensions/repository/view/captcha http://typo3.org/extensions/repository/view/sr_freecap http://typo3.org/extensions/repository/view/spamshield	Verschiedene Extensions
	Sitzungsverwaltung	CSRF-Schutz	(0)	http://buzz.typo3.org/teams/security/article/typo3-45-will-be-the-most-secure-typo3-version-ever/	Schutz durch Token, das bei jeder Anfrage mitgesendet wird.
		Secure- und HttpOnly-Flag bei Cookies	(0)	http://typo3.org/extension-manuals/doc_guide_security/1.0.1/view/1/7/#id1204049	Option für secure-flag (cookieSecure).
Service Transition					
Installation / Deployment					
Vorbedingungen / Anforderungen	Root Access auf System		(+)	http://wiki.typo3.org/TYPO3_Installation_Basics#How_to_install	Root Access nicht notwendig.
	Transparenz der Abhängigkeiten; gibt es ein Build-System wie maven o.ä.		(-)	Testinstallation, Dokumentation	Standalone-Applikation in PHP; erfordert kein Build-System
	Bibliotheken integriert, die möglicherweise Konflikte auslösen		(+)	Testinstallation	Es werden keine zusätzlichen Bibliotheken mitgeliefert oder vorhandene verändert.

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
Komplexität	Anzahl manueller Vor-/Nacharbeiten bei Konfiguration und Installation		(-)	http://wiki.typo3.org/Security#TYPO3_Security, Testinstallation	Mehrere Einstellungen müssen angepasst werden, Zugriff auf sicherheitsrelevante Dateien und Verzeichnisse muss eingeschränkt werden, Account und Passwörter müssen gelöscht oder geändert werden.
	Verwendung von Standards	Automatisiertes Update durch Paketverwaltung	(-)	http://typo3.org/news/article/typo3-60-back-to-the-future/	Geplant für 6.1
	Sichere Default-Installation		(-)	http://typo3.org/news/article/typo3-60-back-to-the-future/	Die Default-Installation ist in einigen Punkten unsicher: cookieSecure = 0 (always send cookie) cookieHttpOnly = "" lockSSL = 0 (Backend auch über http statt nur https) Alle Standardwerte sind allgemein bekannt.
	Hinweis auf sicherheitsrelevante Einstellungen an Ort und Stelle		(-)		Kaum Hinweise in Konfigurationsdateien
	Übernahme sicherheitsrelevanter Einstellungen bei Updates		(+)	http://wiki.typo3.org/Upgrade	Upgrade beeinflusst Einstellungen in Datenbank nicht
Anwenderdokumentation					
Lieferung von Guidelines, Tutorials			(+)	1) http://typo3.org/documentation/document-library/tutorials//current/ 2) http://typo3.org/teams/security/resourc	1) Tutorials 2) Educational Resources 3) Blog des Security Teams

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				es/ 3) http://buzz.typo3.org/teams/security/	
Qualität	Zielgruppenorientierung		(+)	http://wiki.typo3.org/Main_Page	Das Wiki ist unterteilt für Entwickler, Administratoren und Editoren.
	Navigation und Suche		(+)	Test	Aufbau als Wiki - Informationen stehen direkt am Schlagwort.
	Mehrsprachigkeit		(-)	Dokumentation bspw. http://www.typo3-handbuch.net/ oder auf der offiziellen Site: http://typo3.org/documentation/document-library/tutorials/doc_tut_quickstart_de/0.0.6/view/	Offizielle Dokumentation nur englisch. Nur Handbuch von Drittanbietern auf deutsch verfügbar
	Aktualität		(-)	http://wiki.typo3.org/Configuration http://wiki.typo3.org/User_Management http://typo3.org/documentation/document-library/doc_tut_editor_ger/	Einige Dokumentation ist als „Outdated“ gekennzeichnet. Andere wichtige Dok. ist unvollständig oder provisorisch oder gar nicht vorhanden (404) Auch viele Security Docs sind über ein Jahr alt Schon die Installations-Doku enthält kaum Versionsangaben zu PHP, MySQL etc.
	Vollständigkeit	Beschreibung aller Sicherheits-mechanismen mit Annahmen und Einschränkungen	(+)	http://docs.typo3.org/typo3cms/SecurityGuide/	Der Security Guide beschreibt typische Risiken und Bedrohungen und gibt Hinweise, wie eine TYPO3 Site geschützt werden kann.

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		Beschreibung aller sicherheits-relevanten Einstellungen	(+)	http://docs.typo3.org/typo3cms/SecurityGuide/	
		Beschreibung verschiedener Szenarien (z.B. externer ID-Provider)	(0)	http://docs.typo3.org/typo3cms/SecurityGuide/	Keine Beschreibung von Szenarien gefunden
Service Operation					
Marktdurchdringung					
Internetpräsenz	Moderierte Foren mit Fokus Security (Anzahl)		(-)	www.typo3forum.net www.typo3.net/forum	Mehrere Foren u.a. zu Installation allgemein. Aber: http://www.typo3forum.net/: keine Kategorie Security, http://www.typo3.net/forum/ auch nicht. Selbst bei forum.typo3.org nicht.
	Akteure mit Fokus Security (Anzahl)		(+)	http://typo3.org/teams/security/members/	12 Mitglieder hat das Security Team
Anzahl Installationen in Deutschland			(+)	http://typo3.org/home/typo3-in-numbers/	1,51% von 7.496.183 de-Domains nutzen TYPO3: 113.191 TYPO3-Installationen.
Behandlung von Änderungen					
Security Patches	Zeit bis zum Erscheinen		(nf)[101]	http://typo3.org/teams/security/security-bulletins/	I.d.R. werden die Schwachstellen erst mit dem Erscheinen des Patches veröffentlicht, es konnte also keine zeitliche Bewertung gefunden/vorgenommen werden.

101 (nf) – nicht gefunden, es wurden keine Informationen hierzu gefunden, eine negative Bewertung erscheint aber nicht angemessen

TYPO3 CMS 6.0					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					How long was the average delay between the date when you got to know the hole and the bug fix? This really depends on the issue, the severity, ... The time is between a day and some weeks. (Georg Ringer, Member of the TYPO3 Security Team)
	Transparenz der Begleitinformationen		(+)	http://forge.typo3.org/projects/typo3v4-documentation/activity	Releases haben Changelog; "Forge" zeigt aktuelle Issues und was bearbeitet wird.
Betrieb					
Logging	Revisionsfähigkeit	integritätsgeschützt	(-)		Keine Informationen hierzu auffindbar.
		vollständig	(0)	Testinstallation	Keine Zuordnung von Aktionen zu Sitzungen (nur zu Accounts).
	Loglevel konfigurierbar		(-)		Keine Informationen hierzu auffindbar.
	Vertraulichkeit	Fachliche Logs von technischen Logs trennbar	(-)	Testinstallation	Nur über Filter bei der Anzeige
		Verschlüsselung der fachlichen Logs mit Bordmitteln möglich	(-)	Testinstallation	sys_log Tabelle in Datenbank speichert alle Aktionen im Typo3-Backend. Technische Administratoren können diese einsehen.
		Rücksicherung offenbart keinen Klartext	(-)	http://typo3.org/extensions/repository/view/w4x_backup	Der Link zeigt eine übliche Extension für vollständige Typo3 Backups. In der Dokumentation ist nicht von einer Verschlüsselung zu lesen.

TYPO3 CMS 6.0				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Einbindung in System-management	Laufzeitinformationen extrahierbar	(-)		Es wurden keine Informationen hierzu gefunden.
	Steuerung per Skript	(-)		Keine Skriptfähigkeit, keine Hinweise gefunden.
	Steuerung per API	(+)	http://typo3.org/api/typo3cms/class_typo3_1_1_cms_1_1_core_1_1_core_1_1_bootstrap.html	Die API-Klasse Bootstrap.php hat Public Member Functions BaseSetup und Shutdown.

Anhang: Ergebnistabelle WordPress 3.4.2

WordPress 3.4.2				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Design				
Geschäftsführung (Governance)				
Lizenzierung	Ausprobierbarkeit	(+)	http://www.opensourcecms.com/scripts/details.php?scriptid=88	Drittanbieter (vendor-independant resource for Open Source CMS)
Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen		(+)	http://codex.wordpress.org/Main_Page http://codex.wordpress.org/Developer_Documentation http://codex.wordpress.org/Hardening_WordPress	
Transparenz der Kommunikation bei Schwachstellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	(0)	1) http://automattic.com/security/ 2) plugins@wordpress.org 3) security@wordpress.org	1) für Sites gehostet durch wordpress.com 2) Plugins betreffend 3) selbst-gehostete Sites
	Bug-Tracking-System mit Kategorie Sicherheit	(np)	http://core.trac.wordpress.org/ http://plugins.trac.wordpress.org/report/1	Trac verfügbar; allerdings keine explizite Security-Kategorie. Bei Security Issues wird auf den direkten Kontakt zu WordPress verwiesen. Dies war im Rahmen der Studie nicht prüfbar.
	Advisories auf der Website des Herstellers leicht auffindbar	(0)	http://wordpress.org/news/category/security/	

WordPress 3.4.2				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
	Beschreibung des Prozesses zur Bearbeitung von Sicherheitsproblemen	(0)	http://codex.wordpress.org/Security_FAQ	Es gibt Hinweise zum Melden von Security Issues, jedoch keine Beschreibung, wie das Security-Team damit verfährt ->schlechte Transparenz
Sicherer Entwicklungsprozess	Einheitliche Qualitätskriterien	(+)	http://codex.wordpress.org/WordPress_Coding_Standards	Coding Guidelines
	Aussagekräftige und durchgängige Kommentierung im Quelltext	(+)	http://core.svn.wordpress.org/trunk/	
	Strukturelle Mittel zur Durchsetzung der Qualitätskriterien	(+)	http://codex.wordpress.org/How_does_code_make_it_into_WordPress http://codex.wordpress.org/Reporting_Bugs	Patches werden vom Developer Team angenommen oder abgewiesen.
	Transparenz des Entwicklungsprozesses	(+)	http://core.trac.wordpress.org/	
Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen		(+)	http://codex.wordpress.org/Hardening_WordPress	Allgemeine Sicherheitsdokumentation.
Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene		(-)		Hierzu konnten leider im angemessenen Zeitrahmen keine Dokumentationen gefunden werden.
Prüfung / Test (Verifikation)				
Design-Review nachweislich		(-)	http://codex.wordpress.org/How_does_code_make_it_into_WordPress	Die Haupt-Entwickler von WordPress haben als Einzige die Möglichkeit Code in das Produkt einzufügen. Es kann vermutet werden, dass dies nach einem Review geschieht. Jedoch

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					ist dies nicht transparent.
Code-Review nachweislich			(-)	http://codex.wordpress.org/How_does_code_make_it_into_WordPress	Nicht transparent
Sicherheitstests nachweislich			(-)		Nicht transparent
Softwareentwicklung (Construction)					
Sichere Architektur	Modularisierung / Trennung der logischen Funktionen	Modularisierung / Trennung der logischen Funktionen	(0)	http://codex.wordpress.org/Developer_Documentation	Verschiedene APIs; Plugin-Schnittstelle WordPress benutzt das LAMP- bzw. WAMP Prinzip (Linux/Windows, Apache, MySQL, PHP). Die einzelnen Module können bei Bedarf mit gleichwertiger Software ersetzt werden. Die Funktionen sind entsprechend logisch auf die Module aufgeteilt.
		Trennung in verschiedene Sicherheitsniveaus möglich (Sandbox)	(0)	http://codex.wordpress.org/Test_Driving_WordPress	In Kombination mit lokaler Konfiguration eines Webservers und einer MySQL-Datenbank ist Sandboxing für Testzwecke möglich. Für Produktion kann man nur auf Betriebssystemmittel zurückgreifen.
	Basistechnologie	Verwendung sicherer Bibliotheken	(+)	Testinstallation	Bei einer Standardinstallation werden keine zusätzlichen Bibliotheken zur verwendeten PhP-Version installiert.
		Ausrichtung an relevanten Standards	(+)	Testinstallation	Die genannten Punkte können alle mit "JA" beantwortet werden. Einige davon sind jedoch vom Template abhängig. Beispiel: Entwickler1 gestaltet sein Layout mit jQuery, Entwickler2 gestaltet sein Layout mit Mootools.
	Integrations-	Web Services	(0)	http://wordpress.org/extend/plugins/wor	Plugins verfügbar

WordPress 3.4.2					
Kriterium			*Wert*	*Quelle*	*Anmerkung*
	fähigkeit			dpress-web-service/ http://wordpress.org/extend/plugins/soap-authentication/	
		Payment	(+)	http://wordpress.org/extend/plugins/search.php?q=payment	Realisiert über Plugins; PayPal- und Invoice-Plugins.
		Authentifizierung / Autorisierung	(+)	http://wordpress.org/extend/plugins/openid/ http://wordpress.org/extend/plugins/oauth-provider/ http://wordpress.org/extend/plugins/simple-ldap-login/	OpenID, OAuth sowie LDAP.
	Skalierbarkeit des Systems	Nutzung von Caching Mechanismen	(+)	http://codex.wordpress.org/WordPress_Optimization/Caching	
		Verteilbarkeit kritischer Komponenten zur Erhöhung Performance/ Verfügbarkeit	(-)	http://codex.wordpress.org/WordPress_Optimization/Caching	Verteilbarkeit nicht direkt möglich; Performance-Optimierung möglich mit verschiedenen Caching-Plugins und Einstellungen
	Auswahl und Reife der Erweiterungsmechanismen		(0)	http://codex.wordpress.org/Plugin_API http://codex.wordpress.org/Writing_a_Plugin	Erweiterungen sind durch Plugins realisiert die von dritten Parteien entwickelt werden. Die Pflegequalität der Plugins ist dabei sehr unterschiedlich. Die Trennung der APIs nach Er-

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					weiterungsaspekten war im Rahmen der Studie nicht prüfbar.
Umsetzung von Sicherheitsan-forderungen	Rollen und Rechtekonzept	Vererbung von Rechten	(+)	http://codex.wordpress.org/Roles_and_Capabilities	Anzahl der Gruppen ist jedoch beschränkt auf Abonnent, Administrator, Autor, Mitarbeiter, Redakteur
		Definition von Gruppen	(-)	http://codex.wordpress.org/Roles_and_Capabilities	Pre-defined Roles
		Rollen-basierter Zugriff	(+)	http://codex.wordpress.org/Roles_and_Capabilities	
		Attribut-basierter Zugriff	(-)	http://codex.wordpress.org/Roles_and_Capabilities	Zugriffsrechte nur über Rollen definierbar
		Granularität der Zugriffssteuerung	(-)		Inhaltsbereiche lassen sich nicht built-in mit Rollen versehen.
	Anpassbarkeit	Unterstützung mobiler Endgeräte	(+)	http://codex.wordpress.org/Theme_Development http://www.wpbeginner.com/wp-tutorials/11-ways-to-create-a-mobile-friendly-wordpress-site/	Eigene Themes können responsive gestaltet und umgesetzt werden. Es gibt außerdem Plugins für das korrekte Gestalten einer mobilen Version der Seite
		Nicenames / URL-rewriting	(+)	http://codex.wordpress.org/Using_Permalinks	Permalinks konfigurierbar; benutzt mod_rewrite
		Personalisierung	(+)	http://codex.wordpress.org/Theme_Development http://codex.wordpress.org/Writing_a_Plugin	Themes können entwickelt/installiert werden Plugins können entwickelt/installiert werden

WordPress 3.4.2					
Kriterium			*Wert*	*Quelle*	*Anmerkung*
				http://codex.wordpress.org/Plugin_API	
		Kontext-Sensitivität	(-)		Es scheint kein Standardverfahren für Kontextsensitive Inhalte zu geben. Es konnten keine Informationen hierzu gefunden werden.
	Sprachvarianten	Internationalisierung nach Best Practices realisiert	(+)	Testinstallation	UTF-8 in MySQL konfigurierbar; Zeichensatz lässt sich in WordPress-Einstellungen umstellen.
	Sichere Datenablage	Ablage von Zugangsdaten	(+)	Testinstallation	phpass, in Datenbanktabelle
		Rücksicherungen konzeptionell vorbereitet?	(+)	http://codex.wordpress.org/WordPress_Backups	Für WordPress Site und WordPress Database. Optionen für automatisches Backup.
		Vertraulichkeit und Integrität von Benutzerdaten bzw. Vorgangsdaten schützbar?	(-)	http://codex.wordpress.org/Database_Description	Benutzer- und Vorgangsdaten liegen in MySQL-Datenbank. Besonders vertrauliche Daten können nicht mit WordPress selbst geschützt werden.
		Systemdaten von fachlichen Daten separiert	(-)	http://codex.wordpress.org/Database_Description	Fachliche Daten sind systemtechnisch nicht separiert.
	Suchmaschine	Reife der Suchmaschine	(+)	http://wordpress.org/extend/plugins/solr-for-wordpress	
	Inhaltsverwaltung	Redaktioneller Workflow	(+)	http://codex.wordpress.org/Roles_and_Capabilities	Verschiedene Rollen verfügbar.

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		definierbar			
		Versionierung von Inhalten	(+)	http://en.support.wordpress.com/posts/post-revisions/	Revision-Log gespeichert; Zurücksetzen auf alte Version möglich.
		Wiederverwendung von Content zur Reduktion der Komplexität	(+)	Testinstallation	Es gibt eine "Medienverwaltung".
		Preview verfügbar	(+)	Testinstallation	
	Authentifizierung	Registrierung	(+)	Testinstallation	Keine besonderen Schwachstellen erkennbar.
		Anmeldung	(-)	Testinstallation	Offenbart Existenz von Benutzernamen.
		Brute-Force-Schutz (Passwort)	(-)	http://www.frameloss.org/2011/07/29/stopping-brute-force-logins-against-wordpress/	Mit mod_security realisierbar
		Passwort vergessen	(0)	http://codex.wordpress.org/Resetting_Your_Password	
		Captcha	(+)	http://wordpress.org/search/captcha	
	Sitzungsverwaltung	CSRF-Schutz	(0)	http://codex.wordpress.org/Function_Reference/wp_nonce_field http://wordpress.org/extend/plugins/bulletproof-security/	Built-in wp_nonce_field Funktion und verschiedene Plugins verfügbar
		Secure- und HttpOnly-Flag bei Cookies	(0)	http://codex.wordpress.org/WordPress_Cookies	Secure-Flag eingebaut. Cookie-Funktion kann von Plugins ersetzt werden.

WordPress 3.4.2				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Transition				
Installation / Deployment				
Vorbedingungen / Anforderungen	Root Access auf System	(+)	http://codex.wordpress.org/Installing_WordPress#Famous_5-Minute_Install	Root-Rechte unnötig, wenn LAMP Stack schon auf einem Server vorkonfiguriert ist.
	Transparenz der Abhängigkeiten; gibt es ein Build-System wie maven o.ä.	(-)	http://codex.wordpress.org/Installing_WordPress	Standalone-Applikation in PHP; erfordert kein Build-System
	Bibliotheken integriert, die möglicherweise Konflikte auslösen	(+)	Testinstallation	Es werden keine zusätzlichen Bibliotheken mitgeliefert oder vorhandene verändert.
Komplexität	Anzahl manueller Vor-/Nacharbeiten bei Konfiguration und Installation	(+)	http://codex.wordpress.org/Installing_WordPress	Manuelles Bearbeiten des wp_config.php-Files: Datenbankanbindung eintragen.
	Verwendung von Standards: Automatisiertes Update durch Paketverwaltung	(+)	http://codex.wordpress.org/Updating_WordPress#Automatic_Update	Verfügbar ab Version 2.7+; Automatisches Update lässt sich in den Einstellungen aktivieren
	Sichere Default-Installation	(-)	http://codex.wordpress.org/Installing_WordPress http://codex.wordpress.org/Hardening_WordPress Testinstallation	Kein aktivierbarer https-Zugang für Backend.
	Hinweis auf sicherheitsrelevante Einstellungen an Ort und Stelle	(+)	http://core.trac.wordpress.org/browser/trunk/wp-config-sample.php	Konfigurationsdatei wp-config.php ausführlich dokumentiert und kommentiert hinsichtlich Sicherheit.
	Übernahme sicherheitsrelevanter Einstellungen bei Updates	(+)	http://codex.wordpress.org/Updating_WordPress	Konfiguration bleibt erhalten; Plugins weiterhin integriert, sofern mit neuer Version

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					kompatibel
Anwenderdokumentation					
Lieferung von Guidelines, Tutorials			(-)	http://codex.wordpress.org/Main_Page http://learn.wordpress.com/	wenig Sicherheitsrelevantes
Qualität	Zielgruppenorientierung		(+)	http://codex.wordpress.org/Getting_Started_with_WordPress	Verschiedene Themengruppen im Wiki: How to use, Development, Write a Plugin
	Navigation und Suche		(+)	http://codex.wordpress.org/Main_Page http://codex.wordpress.org/Codex:Quick_index http://codex.wordpress.org/Special:AllPages	Codex ist ein Wiki
	Mehrsprachigkeit		(-)	http://codex.wordpress.org/Hardening_WordPress	Bisher nicht in Deutsch verfügbar
	Aktualität		(+)	http://codex.wordpress.org/Special:RecentChanges	Regelmäßige Updates des WordPress Codex
	Vollständigkeit	Beschreibung aller Sicherheitsmechanismen mit Annahmen und Einschränkungen	(+)	http://codex.wordpress.org/Hardening_WordPress	Dokument Hardening WordPress
		Beschreibung aller sicherheitsrelevanten Einstellungen	(+)	http://codex.wordpress.org/Hardening_WordPress	

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		Beschreibung verschiedener Szenarien (z.B. externer ID-Provider)	(+)	http://codex.wordpress.org/FAQ_Advanced_Topics#Alternate_Uses_of_WordPress	Einsatzmöglichkeiten von WordPress als CMS, Multi-Blog etc.
Service Operation					
Marktdurchdringung					
Internetpräsenz	Moderierte Foren mit Fokus Security (Anzahl)		(-)	http://wordpress.org/support/	Kein explizites Security-Forum, allerdings Bereiche mit Sicherheitsaspekten.
	Akteure mit Fokus Security (Anzahl)		(+)	http://codex.wordpress.org/Version_3.5.1	Ein "WordPress security team" wird erwähnt, in früheren Versionen: "WordPress Core Security Team".
Anzahl Installationen in Deutschland			(+)	http://en.wordpress.com/stats/	Weltweit fast 60 Millionen Installationen
Behandlung von Änderungen					
Security Patches	Zeit bis zum Erscheinen		(-)	http://codex.wordpress.org/Hardening_WordPress#Security_Themes http://core.trac.wordpress.org/ticket/11819	I.d.R. werden die Schwachstellen erst mit dem Erscheinen des Patches veröffentlicht. Ein wieder eröffnetes Security-Ticket war nominell insgesamt drei Jahre offen (Link 2)
	Transparenz der Begleitinformationen		(-)	http://codex.wordpress.org/WordPress_Versions http://plugins.trac.wordpress.org/report/1	Changelogs und zusätzliche Informationen verfügbar, aber keine Einschätzung zum Risiko. Nicht erkennbar, ob es sich um Security Issues handelt oder andere Bugs.
Betrieb					
Logging	Revisionsfähigkeit	Integritätsge-	(-)	http://codex.wordpress.org/Editing_wp-c	Nur PHP-Error Logging

WordPress 3.4.2					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		schützt		onfig.php#Debug	
		Vollständig	(-)	http://codex.wordpress.org/Editing_wp-config.php#Debug	Nur PHP-Error Logging
	Loglevel konfigurierbar		(0)	http://codex.wordpress.org/Editing_wp-config.php#Debug	Ist eher als PHP Debug Log zu sehen
	Vertraulichkeit	Fachliche Logs von technischen Logs trennbar	(-)	http://codex.wordpress.org/Debugging_in_WordPress#PHP_Errors.2C_Warnings_and_Notices	Debug-Funktionalität ist nur technischer Log von PHP Errors (WP_DEBUG)
		Verschlüsselung der fachlichen Logs mit Bordmitteln möglich	(-)	http://codex.wordpress.org/Debugging_in_WordPress	Debug-Funktionalität wird nur für lokale Installationen empfohlen
		Rücksicherung offenbart keinen Klartext	(+)	http://ithemes.com/backupbuddy-stash/	Nur über Plugin - Backupbuddy ist bspw. ein beliebtes Plugin für WordPress-Backups und verschlüsselt die Sicherung.
Einbindung in System-management	Laufzeitinformationen extrahierbar		(-)		Systemmanagement in dieser Form nicht möglich.
	Steuerung per Skript		(-)		Nicht skriptfähig. Keine Hinweise hierzu gefunden.
	Steuerung per API		(+)	http://codex.wordpress.org/Settings_API	

Anhang: Ergebnistabelle Joomla! 3.02

Joomloa! 3.02				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Design				
Geschäftsführung (Governance)				
Lizensierung	Ausprobierbarkeit	(+)	http://demo.joomla.org/	
Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen		(+)	http://docs.joomla.org/Secure_coding_guidelines	
Transparenz der Kommunikation bei Schwach-stellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	(0)	http://developer.joomla.org/security.html	
	Bug-Tracking-System mit Kategorie Sicherheit	(-)		Nicht gefunden
	Advisories auf der Website des Herstellers leicht auffindbar	(0)	http://docs.joomla.org/Security	Diverse Checklists, FAQs und How-Tos; Für Erweiterungen: Vulnerable Extensions List.
	Beschreibung des Prozesses zur Be-arbeitung von Sicherheitsproblemen	(0)	http://docs.joomla.org/Security_Checklist/You_have_been_hacked_or_defaced	
Sicherer Ent-wicklungsprozess	Einheitliche Qualitätskriterien	(+)	http://developer.joomla.org/5-policies/3-Joomla-Coding-Standards.html und http://joomla.github.com/joomla-platform/	Coding guidelines
	Aussagekräftige und durchgängige Kommentierung im Quelltext	(+)	http://joomlacode.org/svn/joomla/development/trunk/	

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Strukturelle Mittel zur Durchsetzung der Qualitätskriterien		(+)	http://joomla.github.com/joomla-platform/	Menüpunkt Code Analysis
	Transparenz des Entwicklungs-prozesses		(+)	http://joomlacode.org/gf/project/joomla/	
Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen			(-)		Keine Informationen hierzu verfügbar
Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene			(-)		Keine Informationen hierzu verfügbar
Prüfung / Test (Verifikation)					
Design-Review nachweislich			(-)		Keine Informationen hierzu verfügbar
Code-Review nachweislich			(-)		Nur für Security-Patches
Sicherheitstests nachweislich			(-)		Keine Informationen hierzu verfügbar
Softwareentwicklung(Construction)					
Sichere Architektur	Modularisierung / Trennung der logischen Funktionen	Modularisierung / Trennung der logischen Funktionen	(0)	Testinstallation	Content Server und Auslieferungssystem sind nicht trennbar. Joomla! benutzt aber das LAMP- bzw. WAMP Prinzip (Linux/Windows, Apache, MySQL, PHP). Die einzelnen Module können bei Bedarf mit gleichwertiger Software ersetzt werden. Die Funktionen sind entsprechend logisch auf die Module aufgeteilt.
		Trennung in verschiedene Sicherheitsniveaus möglich (Sandbox)	(0)	Testinstallation	Keine CMS Funktionalität verfügbar, Zurückgreifen auf OS Mittel.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Basistechnologie	Verwendung sicherer Biblio-theken	(+)	Testinstallation	Bei einer Standardinstallation werden keine zusätzlichen Bibliotheken zur verwendeten PhP-Version installiert.
		Ausrichtung an relevanten Standards	(+)	Testinstallation	XHTML (core), jquery (core), CMIS (Modul)
	Integrations-fähigkeit	Web Services	(0)	http://extensions.joomla.org/extensions/miscellaneous/development/21684 http://www.joomlaos.de/Downloads/Joomla_und_Mambo_Addons/Universeller_Application_Connector.html	SOAP (extension)
		Payment	(0)	http://extensions.joomla.org/extensions/e-commerce/payment-systems	
		Authentifizierung/ Autorisierung	(0)	http://www.joomla.org/core-features.html	LDAP, OpenID
	Skalierbarkeit des Systems	Nutzung von Caching Mechanismen	(+)	http://docs.joomla.org/Cache	
		Verteilbarkeit kritischer Komponenten zur Erhöhung Per-formance/ Ver-fügbarkeit	(-)	Testinstallation	Standardmäßig keine Verteilung der Komponenten vorgesehen.
	Auswahl und Reife der Er-		(0)	http://docs.joomla.org/Developers	Definierte Schnittstellen, aber keine Trennung.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	weiterungsmechanismen			http://docs.joomla.org/Plugin_Development http://docs.joomla.org/Module_Development	Es gibt noch keine brauchbare Dokumentation für Joomla! 3.
Umsetzung von Sicherheits-anforderungen	Rollen und Rechtekonzept	Vererbung von Rechten	(+)	http://docs.joomla.org/Access_Control_List/2.5/Tutorial#Actions.2C_Groups.2C_and_Inheritance	
		Definition von Gruppen	(+)	http://www.joomla.org/core-features.html http://docs.joomla.org/Access_Control_List/2.5/Tutorial#Default_ACL_Setup	Neun vordefinierte Usergruppen (Version 1.5). Unbegrenzt in Version 2.5.
		Rollen-basierter Zugriff	(+)	Testinstallation	"roles" und "groups" werden synonym verwendet.
		Attribut-basierter Zugriff	(-)		Ein attributsbasierter Zugriff ist nicht dokumentiert.
		Granularität der Zugriffssteuerung	(+)	http://docs.joomla.org/Access_Control_List/2.5/Tutorial#Default_Groups	
	Anpassbarkeit	Unterstützung mobiler Endgeräte	(+)	http://www.joomla.org/3/en	mobile r3ady
		Nicenames / URL-rewriting	(+)	Testinstallation	SEF/SEO URLs
		Personalisierung	(-)		Es konnten keine Informationen darüber gefunden werden.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		Kontext-Sensitivität	(-)		Es konnten keine Informationen darüber gefunden werden.
	Sprachvarianten	Internationalisierung nach Best Practices realisiert	(+)	http://www.joomla.org/core-features.html	Demo Templates werden als UTF8 ausgeliefert
	Sichere Datenablage	Ablage von Zugangsdaten	(-)	Testinstallation	Benutzt MD5-Hash mit Salt in Datenbanktabelle, laut aktueller Liste der Bundesnetzagentur nicht geeignet.
		Rücksicherungen konzeptionell vorbereitet?	(-)		Datenbankablage; Wiederherstellung einzelner Inhaltsbereiche sind nicht dokumentiert.
		Vertraulichkeit und Integrität von Benutzerdaten bzw. Vorgangsdaten schützbar?	(-)	http://docs.joomla.org/Database_structure http://www.torkiljohnsen.com/wp-content/uploads/2006/04/joomla_15_database_schema.png	Gemeinsame Datenbank.
		Systemdaten von fachlichen Daten separiert	(0)	http://docs.joomla.org/Database_structure http://www.torkiljohnsen.com/wp-content/uploads/2006/04/joomla_15_database_schema.png	Separate Tabellen in der DB.
	Suchmaschine	Reife der Suchmaschine	(+)	http://www.solrhq.com/solrhq-for-cms/joomla-extension-for-solr-and-solrhq/	Wenig Infos, aber es gibt ein Joomla-internes Search-Modul und eine Solr Anbindung
	Inhaltsverwaltung	Redaktioneller Workflow	(0)	Testinstallation http://www.joomlaos.de/Downloads/Jooml	Kein eingebauter Workflow. Rudimentär über Erweiterungen wie bspw. „Workflow“.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		definierbar		a_und_Mambo_Komponenten/Workflow.html	
		Versionierung von Inhalten	(+)	http://extensions.joomla.org/extensions/authoring-a-content/content-versioning/6260	Als Component nutzbar
		Wiederver-wendung von Content zur Reduktion der Komplexität	(+)	Testinstallation	
		Preview verfügbar	(+)	Testinstallation	
	Authenti-fizierung	Registrierung	(+)	Testinstallation	Keine besonderen Schwachstellen erkennbar.
		Anmeldung	(0)	http://extensions.joomla.org/extensions/access-a-security/site-security/login-protection/11519	Wenn keine SSL-Verbindung benutzt wird, schickt ein Nutzer seine Daten im Klartext. Das verlinkte Plugin nutzt RSA, um diese Daten zu verschlüsseln. Keine sonstigen Schwachstellen erkennbar.
		Brute-Force-Schutz (Passwort)	(0)	http://extensions.joomla.org/extensions/access-a-security/site-security/login-protection/22982	Externes Plugin Brute Force Stop, blockiert nicht Nutzer, sondern IP-Adressen, Konfigurationsseite nicht selbsterklärend.
		Passwort vergessen	(0)	Testinstallation	Verification code per email.
		Captcha	(+)	Beispielplugin: http://extensions.joomla.org/extensions/access-a-security/site-security/captcha/20938	Über Plugins.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Sitzungs-verwaltung	CSRF-Schutz	(0)	http://docs.joomla.org/How_to_add_CSRF_anti-spoofing_to_forms	Schutz durch Token (Zufallsstring), das bei relevanten POST und GET Anfragen mit-gesendet wird.
		Secure- und HttpOnly-Flag bei Cookies	(0)	http://docs.joomla.org/JInputCookie::set/11.1	
Service Transition					
Installation / Deployment					
Vorbedingungen / Anforderungen	Root Access auf System		(+)	http://docs.joomla.org/Installing_Joomla!_3.0	Root-Rechte nicht erforderlich für Installation der PHP-Dateien und für Datenbankzugriff
	Transparenz der Abhängigkeiten; gibt es ein Build-System wie maven o.ä.		(-)	http://docs.joomla.org/Installing_Joomla!_3.0	Beim Setup wird nur das Vorhandensein einiger erforderlicher Komponenten geprüft. Welche Pakete genutzt werden, ist nicht er-sichtlich.
	Bibliotheken integriert, die mög-licherweise Konflikte auslösen		(+)	Testinstallation	Es werden keine zusätzlichen Bibliotheken mitgeliefert oder vorhandene verändert.
Komplexität	Anzahl manueller Vor-/Nacharbeiten bei Konfiguration und Installation		(+)	http://docs.joomla.org/Installing_Joomla!	Alles Teil der geführten Installation
	Verwendung von Standards	Automatisiertes Update durch Paketverwaltung	(+)	http://docs.joomla.org/Joomla_Update_System	Joomla! Update Manager. Muss manuell aus-gelöst werden, installiert dann aber auto-matisch. Wenig dokumentiert.
	Sichere Default-Installation		(-)	Testinstallation	Backend-Zugang standardmäßig ohne https
	Hinweis auf sicherheitsrelevante Einstellungen an Ort und Stelle		(-)	configuration.php	Datei configuration.php-dist hatte einige Hinweise, die vom Setup erzeugte Datei nicht.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Übernahme sicherheitsrelevanter Einstellungen bei Updates		(+)		Konfigurationsdateien bleiben erhalten
Anwenderdokumentation					
Lieferung von Guidelines, Tutorials			(+)	http://docs.joomla.org/ http://docs.joomla.org/Security_and_Performance_FAQs Newsfeed:Joomla! Developer Network - Security News	Vielzahl von Dokumenten mit Security Bezug
Qualität	Zielgruppenorientierung		(+)	http://docs.joomla.org/	Reader profiles: Beginners, developers, web designers, administrators, evaluators und weitere acht Rollen.
	Navigation und Suche		(+)	http://docs.joomla.org/	Gut strukturiert; besonders die zielgruppenspezifischen Quick Links sind hilfreich.
	Mehrsprachigkeit		(0)	http://www.joomla-security.de/	Das Projekt Joomla! Security hat begonnen, deutschsprachige Dokumente zum Thema Sicherheit bereitzustellen.
	Aktualität		(+)	http://docs.joomla.org/	Einige Dokumente sind noch nicht für die aktuelle Version 3.0 verfügbar, aber bis auf einzelne Ausnahmen sind die Dokumente für Version 2.5 weiterhin anwendbar.
	Vollständigkeit	Beschreibung aller Sicherheits-mechanismen mit Annahmen und	(0)	http://docs.joomla.org/Security	Die aufgelisteten Guidelines und FAQs enthalten Beschreibungen einiger Mechanismen (z.B. Windows/Unix File Permissions), sind aber nicht vollständig und geben keine Hin-

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		Einschränkungen			weise auf Annahmen oder Einschränkungen.
		Beschreibung aller sicherheits-relevanten Ein-stellungen	(0)	http://docs.joomla.org/Security	Es gibt einzelne Beschreibungen jedoch keine zusammenfassende Beschreibung der sicher-heitsrelevanten Einstellungen.
		Beschreibung ver-schiedener Szenarien (z.B. externer ID-Provider)	(-)	http://docs.joomla.org	Keine Beschreibung von Szenarien
Service Operation					
Marktdurchdringung					
Internetpräsenz	Moderierte Foren mit Fokus Security (Anzahl)		(+)	http://docs.joomla.org/Security	Für jede Joomla!-Version gibt es ein eigenes Security-Forum
	Akteure mit Fokus Security (Anzahl)		(+)	http://developer.joomla.org/security.html	Es gibt ein Joomla! Security Strike Team
Anzahl Installationen in Deutschland			(+)	http://w3techs.com/technologies/overview/content_management/all	ca. 1175 Sites in .de nutzen Joomla
Behandlung von Änderungen					
Security Patches	Zeit bis zum Erscheinen		(nf)	http://www.joomla.org/announcements/release-news/5478-joomla-3-0-3-released.html	Behebung dreier (low-priority) Sicherheits-schwachstellen dauerte zwischen zwei Wochen und drei Monaten.
	Transparenz der Begleitinformationen		(+)	http://www.joomla.org/announcements/release-news/5478-joomla-3-0-3-released.html	Liste der Issues und News Features mit detaillierter Beschreibung; Verweis auf eventuelle Probleme nach dem Update.

Joomloa! 3.02					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					Es ist aber nicht ganz klar, ob tatsächlich alle Sites betroffen sind, unabhängig von der Funktionalität, die sie verwenden.
Betrieb					
Logging	Revisionsfähigkeit	Integritätsgeschützt	(-)	http://extensions.joomla.org/extensions/access-a-security/site-security/site-protection/19452	Nicht im System integriert. Die verlinkte Extension kann überprüfen, ob Daten auf dem System verändert wurden. Es wird nicht erwähnt, ob dazu auch die dazugehörigen Logdaten gehören.
		Vollständig	(-)		Keine Informationen über Logging gefunden.
	Loglevel konfigurierbar		(-)		Keine Informationen über Logging gefunden.
	Vertraulichkeit	Fachliche Logs von technischen Logs trennbar	(-)		Keine Informationen über Logging gefunden.
		Verschlüsselung der fachlichen Logs mit Bordmitteln möglich	(-)		Keine Informationen über Logging gefunden.
		Rücksicherung offenbart keinen Klartext	(-)		Keine Informationen über Logging gefunden.
Einbindung in Systemmanagement	Laufzeitinformationen extrahierbar		(-)		Keine Informationen gefunden.
	Steuerung per Skript		(-)		Keine Skriptfähigkeit, keine Hinweise gefunden.

Joomloa! 3.02				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
	Steuerung per API	(-)		Keine Informationen gefunden.

Anhang: Ergebnistabelle Drupal 7.17

Drupal 7.17				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Design				
Geschäftsführung (Governance)				
Lizenzierung	Ausprobierbarkeit	(-)		Es ist zzt. keine offizielle Testinstallation auffindbar. Interessierte müssen selbst eine Testinstallation erstellen.
Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen		(+)	http://drupal.org/documentation/develop	Es gibt ausführliche Beispiele und vorprogrammierte Module. Außerdem gibt es Anleitung wie JavaScript und SQL genutzt werden können.
Transparenz der Kommunikation bei Schwachstellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	(0)	http://drupal.org/node/101494	Sicherheitsprobleme können entweder über die Seite security.drupal.org oder die E-Mail-Adresse security@drupal.org gemeldet werden
	Bug-Tracking-System mit Kategorie Sicherheit	(0)	http://drupal.org/security/psa	Sicherheitsprobleme werden erst veröffentlicht, wenn es einen Patch gibt
	Advisories auf der Website des Herstellers leicht auffindbar	(0)	http://drupal.org/security/contrib	
	Beschreibung des Prozesses zur Bearbeitung von Sicherheitsproblemen	(0)	http://drupal.org/node/213320	
Sicherer Entwicklungsprozess	Einheitliche Qualitätskriterien	(+)	http://drupal.org/coding-standards http://drupal.org/writing-secure-code	

Drupal 7.17				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
	Aussagekräftige und durchgängige Kommentierung im Quelltext	(+)	https://github.com/xeraa/cms-security/blob/master/README.md	In der Quelle wird gezeigt, wie viele Zeilen Quellcode und Kommentare der Core-Build enthält. Das Dokument enthält aber keine Aussage über die Qualität der Kommentare. Stichproben zeigen Aussagekräftige Kommentierung.
	Strukturelle Mittel zur Durchsetzung der Qualitätskriterien	(-)		Keine Informationen hierzu auffindbar.
	Transparenz des Entwicklungsprozesses	(+)	http://drupal.org/project/drupal	Es wird beschrieben, was in den einzelnen Versionen verändert wurde. Dazu gehören auch die evtl. beseitigten Bugs, wodurch diese zustande gekommen sind, wer den Bug entdeckt hat und wer ihn beseitigt hat.
Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen		(+)	http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best?page=0,5	
Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene		(0)	http://drupal.org/node/360052 http://drupalsecurityreport.org/about-drupal-security-report	Die Dokumentation ist vorhanden aber nicht vollständig.
Prüfung / Test (Verifikation)				
Design-Review nachweislich		(+)	http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best?page=0,3	Siehe auch Sicherheitstests. Hängt von der Modulgruppe ab
Code-Review nachweislich		(+)	http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-sec	Siehe auch Sicherheitstests. Hängt von der Modulgruppe ab

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				urity-best?page=0,3	
Sicherheitstests nachweislich			(+)	http://growingventuresolutions.com/blog/security-review-module-and-securing-your-drupal-site.html http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best?page=0,3	Der Kern des Systems gilt als sicher und getestet. Zusätzlich installierte Module scheinen dagegen nicht von einem offiziellen Team getestet zu werden. Für diesen Test gibt es unter dem Link wiederum ein Modul.
Softwareentwicklung (Construction)					
Sichere Architektur	Modularisierung / Trennung der logischen Funktionen	Modularisierung / Trennung der logischen Funktionen	(0)	http://alphanodes.de/high-performance-drupal-infrastrutur	Drupal benutzt das LAMP- bzw. WAMP Prinzip (Linux/Windows, Apache, MySQL, PHP). Die einzelnen Module können bei Bedarf mit gleichwertiger Software ersetzt werden. Die Funktionen sind entsprechend logisch auf die Module aufgeteilt.
		Trennung in verschiedene Sicherheitsniveaus möglich (Sandbox)	(0)	http://drupal.org/node/1011196	Sandbox Projekte sind für Entwicklungszwecke gedacht. Sandbox noch experimentell.
	Basistechnologie	Verwendung sicherer Bibliotheken	(+)		Bei einer Standardinstallation werden keine zusätzlichen Bibliotheken zur verwendeten PhP-Version installiert.
		Ausrichtung an relevanten Standards	(+)	http://drupal.org/project/xhtml	Unterstützung kann mit Modulen nachgerüstet werden. Der Link zeigt ein Modul für xhtml als Beispiel.
	Integrationsfähig-	Web Services	(+)	http://drupal.org/project/soapclient	

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	keit	Payment	(0)	http://drupal.org/project/payment	Ein Payment Modul ist verfügbar
		Authentifizierung/ Autorisierung	(0)	http://drupal.org/project/simple_ldap http://drupal.org/project/oauth2 http://drupal.org/project/openid	Alle Protokolle stehen als Extension zur Verfügung. (OAuth, LDAP sowie OpenID)
	Skalierbarkeit des Systems	Nutzung von Caching Mechanismen	(+)	http://drupal.org/project/boost http://www.cmsmatrix.org/matrix/cms-matrix	
		Verteilbarkeit kritischer Komponenten zur Erhöhung Performance/ Verfügbarkeit	(+)	http://alphanodes.de/high-performance-drupal-infrastrutur	
	Auswahl und Reife der Erweiterungsmechanismen		(+)	http://api.drupal.org/api/drupal	Die Trennung der APIs nach Erweiterungsaspekten war im Rahmen der Studie nicht prüfbar.
Umsetzung von Sicherheitsanforderungen	Rollen und Rechtekonzept	Vererbung von Rechten	(-)	http://drupal.org/node/627332	Es ist nichts über die Vererbung von Rechten im Content-Modell dokumentiert.
		Definition von Gruppen	(-)	http://drupal.org/node/627158	Die Definition von Gruppen wird nicht erwähnt.
		Rollen-basierter Zugriff	(+)	http://www.drupalcenter.de/handbuch/28797	
		Attribut-basierter	(-)		Ein attributbasierter Zugriff ist nicht

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		Zugriff			dokumentiert.
		Granularität der Zugriffssteuerung	(+)	http://drupal.org/getting-started/6/admin/user	Die Dokumentation ist nur sehr kurz.
	Anpassbarkeit	Unterstützung mobiler Endgeräte	(+)	http://drupal.org/documentation/mobile	
		Nicenames / URL-rewriting	(+)	http://codex.gallery2.org/Integration:Drupal:Installation:URL_Rewrite	
		Personalisierung	(-)		Es konnten keine Informationen darüber gefunden werden.
		Kontext-Sensitivität	(+)	http://drupal.org/project/context	
	Sprachvarianten	Internationalisierung nach Best Practices realisiert	(+)	http://www.drupalcenter.de/node/251	
	Sichere Datenablage	Ablage von Zugangsdaten	(+)	http://stackoverflow.com/questions/5031662/what-is-drupals-default-password-encryption-method	Drupal nutzt SHA512 mit Salt und mehreren Iterationen in Datenbanktabelle.
		Rücksicherungen konzeptionell vorbereitet?	(+)	http://drupal.org/project/backup_migrate	
		Vertraulichkeit und Integrität von Benutzerdaten bzw. Vorgangsdaten	(-)	http://drupal.org/project/encrypt_submissions	AES-256, aber nur Transportverschlüsselung.

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		schützbar?			
		Systemdaten von fachlichen Daten separiert	(-)		Hierzu gibt es keine Informationen.
	Suchmaschine	Reife der Such-maschine	(+)	http://drupal.org/documentation/modules/search http://drupal.org/project/search_api_solr_view_modes	
	Verwaltung der Inhalte	Redaktioneller Workflow definierbar	(+)	http://drupal.org/project/workflow_adv	
		Versionierung von Inhalten	(+)	http://drupal.org/project/versioncontrol	
		Wiederverwendung von Content zur Reduktion der Komplexität	(+)	http://drupal.org/project/linodef	
		Preview verfügbar	(+)	http://drupal.org/project/pagepreview	Standardmäßig verfügbar und mit Modulen erweiterbar.
	Authentifizierung	Registrierung	(+)	Testinstallation	Keine besonderen Schwachstellen erkennbar.
		Anmeldung	(+)	Testinstallation	Keine Schwachstellen erkennbar.
		Brute-Force-Schutz (Passwort)	(0)	http://drupal.org/node/117056 http://www.greeniguanamarketing.com/a	Brute-Force-Schutz: im Grundsystem – aber kein Reset vorgesehen. Über eine Extension scheinen wenigstens die Parameter einstellbar

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				rticle/drupal-7-protects-against-brute-force-attacks http://drupal.org/project/flood_control	zu sein. Dies wurde aber nicht geprüft.
		Passwort vergessen	(0)	Testinstallation, http://drupal.org/project/security_questions	One-time-link, 1 Tag gültig, per email. Wenn die Funktion aktiviert ist, wird eine Sicherheitsfrage gestellt. Die Sicherheit des Systems hängt von der ausgewählten Frage und der vom Nutzer festgelegten Antwort ab.
		Captcha	(+)	http://drupal.org/search/site/captcha	
	Sitzungsverwaltung	CSRF-Schutz	(0)		Drupal ist grundsätzlich anfällig für CSRF Angriffe. Es gibt aber zu jeder Lücke eine dokumentierte Lösung, die meist händisch umgesetzt werden muss.
		Secure- und HttpOnly-Flag bei Cookies	(0)	http://www.echoditto.com/blog/dont-share-your-cookies-drupal-and-httponly-flag	
Service Transition					
Installation / Deployment					
Vorbedingungen / Anforderungen	Root Access auf System		(+)	http://drupal.org/documentation/install	
	Transparenz der Abhängigkeiten; gibt es ein Build-System wie maven o.ä.		(-)		In der Beschreibung jedes Moduls sind die Abhängigkeiten dargestellt. Ein Build-System ist nicht zu finden. Aus der Beschreibung der Module kann man

Drupal 7.17				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
				nicht sicher ableiten, welche anderen Module/Plugins benötigt werden und welche Inkompatibilitäten es dabei gibt.
	Bibliotheken integriert, die möglicherweise Konflikte auslösen	(+)	Testinstallation	Es werden keine zusätzlichen Bibliotheken mitgeliefert oder vorhandene verändert.
Komplexität	Anzahl manueller Vor-/Nacharbeiten bei Konfiguration und Installation	(+)	http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best?page=0,2	Die Grundinstallation (core) gilt als sicher. Aber der Zugriff auf die Settings-Datei sollte eingeschränkt und der User #1 entfernt werden.
	Verwendung von Standards: Automatisiertes Update durch Paketverwaltung	(-)	http://www.ehow.com/how_7387345_update-drupal-modules.html	Kann mit cron jobs automatisiert werden.
	Sichere Default-Installation	(+)	http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best?page=0,2	Außer Entfernen von user #1 (s.o.)
	Hinweis auf sicherheitsrelevante Einstellungen an Ort und Stelle	(+)	Testinstallation	
	Übernahme sicherheitsrelevanter Einstellungen bei Updates	(-)	http://drupal.org/node/1223018 siehe UPGRADE.txt	If you made modifications to files like .htaccess or robots.txt, you will need to re-apply them from your backup, after the new files are in place.
Anwenderdokumentation				
Lieferung von Guidelines, Tutorials		(+)	http://drupal.org/documentation	U.a. Administration & Security Guide
Qualität	Zielgruppenorientierung	(+)	http://drupal.org/documentation	User, Developer, Administrators
	Navigation und Suche	(+)	http://drupal.org/documentation	Dokumentations-Einstiegsseite ist gut strukturiert, die einzelnen Themenseiten sind

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					angemessen; die Suche findet die relevanten Dokumente.
	Mehrsprachigkeit		(-)	http://www.drupalcenter.de	drupalcenter.de erarbeitet ein deutsches Benutzerhandbuch
	Aktualität		(+)	http://drupal.org/documentation	Stichproben: aktuell.
	Vollständigkeit	Beschreibung aller Sicherheitsmechanismen mit Annahmen und Einschränkungen	(+)	http://drupal.org/security/secure-configuration	Abschnitt Securing your Site im Administration & Security Guide
		Beschreibung aller sicherheitsrelevanten Einstellungen	(+)	http://drupal.org/security/secure-configuration	
		Beschreibung verschiedener Szenarien (z.B. externer ID-Provider)	(0)	http://drupal.org/security/secure-configuration	Keine Beschreibung von Szenarien
Service Operation					
Marktdurchdringung					
Internetpräsenz	Moderierte Foren mit Fokus Security (Anzahl)		(+)	www.drupal.org/forum www.drupalcenter.de/forum	Drupal.org: Kein spezielles Security-Forum, aber Forum mit den Security Advisories des Security Teams. Drupalcenter: kein Sicherheits-Forum
	Akteure mit Fokus Security (Anzahl)		(+)	https://security.drupal.org/team-members	
Anzahl Installationen in Deutschland			(+)	http://www.handelskraft.de/2011/12/joo	Nicht feststellbar. Am verbreitetsten unter den

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				mla-drupal-wordpress-typo3-und-contao-im-kampf-um-die-spitze/ http://w3techs.com/technologies/overview/content_management/all	top 10.000 Webseiten, allerdings weltweit ca. 690 Websites in .de nutzen Drupal
Behandlung von Änderungen					
Security Patches	Zeit bis zum Erscheinen		(-)	http://drupal.org/security http://drupal.org/SA-CORE-2012-003	Der Vergleich eines Security Fixes, der am 17.10.2012 bekannt gegeben wurde, mit den Quellen brachte nichts, weil einer der beteiligten Security-Team Member einen Branch eingecheckt hat, den er vorher irgendwann gebildet hat, ohne dass man sehen kann, wann er es getan hat.
	Transparenz der Begleitinformationen		(+)	http://drupal.org/SA-CORE-2013-001	Link zeigt ein Beispiel
Betrieb					
Logging	Revisionsfähigkeit	Integritätsgeschützt	(-)		Keine Informationen gefunden.
		Vollständig	(+)	http://drupal.org/project/settings_audit_log	Durch das Modul Audit_Log
	Loglevel konfigurierbar		(+)	http://api.drupal.org/api/drupal/includes!bootstrap.inc/group/logging_severity_levels/7	
	Vertraulichkeit	Fachliche Logs von technischen Logs trennbar	(-)		Keine Informationen gefunden.
		Verschlüsselung	(+)	http://drupal.org/project/encrypt	Es gibt ein Modul, das die Funktion bereit-

Drupal 7.17					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		der fachlichen Logs mit Bordmitteln möglich			stellen kann, falls fachliche Logs möglich sind (s.o.).
		Rücksicherung offenbart keinen Klartext	(+)	http://drupal.org/project/backup_migrate	"AES encryption for backups"
Einbindung in System-management	Laufzeitinformationen extrahierbar		(-)		Keine Informationen gefunden.
	Steuerung per Skript		(+)	http://drupal.org/documentation/modules/dript	Eigene Scriptsprache auf Basis von Lisp
	Steuerung per API		(+)	http://api.drupal.org/api/drupal	

Anhang Ergebnistabelle: Plone 4.21

Plone 4.21				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
Service Design				
Geschäftsführung (Governance)				
Lizenzierung	Ausprobierbarkeit	(+)	http://demo.plone.de/ http://plonedemo.de/ http://plone.org/products/plone/releases/4.2.3	Download und Testinstallation verfügbar
Entwicklerdokumentation/-Schulungen/Anleitungen zu Security-Themen		(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch http://developer.plone.org/index.html# https://buildoutcoredev.readthedocs.org/en/latest/index.html http://docs.zope.org/zope2/zdgbook/Security.html	
Transparenz der Kommunikation bei Schwachstellen	Kontakt des CMS Herstellers zur Meldung von Sicherheitsproblemen	(0)		To report potentially security-related issues, please send a mail to the Plone Security Team at security@plone.org.
	Bug-Tracking-System mit Kategorie Sicherheit	(+)	http://docs.zope.org/zope2/zdgbook/index.html	Extra Meldeweg. Sicherheitslücken sollten nicht öffentlich bekannt sein bevor sie behoben sind. Deshalb werden die Bugs

Plone 4.21				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
				nicht veröffentlicht und nur den entsprechenden Entwicklern bekannt gemacht. Intern vermutlich ein eigenes Bug-Tracking-System. Durchaus ein Sicherheitsgewinn.
	Advisories auf der Website des Herstellers leicht auffindbar	(0)	http://plone.org/products/plone/security/advisories	
	Beschreibung des Prozesses zur Bearbeitung von Sicherheitsproblemen	(0)	http://plone.org/products/plone/security/overview	
Sicherer Entwicklungsprozess	Einheitliche Qualitätskriterien	(+)	https://buildoutcoredev.readthedocs.org/en/latest/style.html https://buildoutcoredev.readthedocs.org/en/latest/index.html http://pypi.python.org/pypi/plone.app.testing/4.2 http://docs.zope.org/zope2/zdgbook/index.html	- Style Guide - Test-driven development
	Aussagekräftige und durchgängige Kommentierung im Quelltext	(+)	https://buildoutcoredev.readthedocs.org/en/latest/intro.html#committing-and-pull-requests	Der ganze Code kann in der zur Verfügung stehenden Zeit nicht auf die Qualität der Kommentare geprüft werden. Entwickler sollen sich an den PEP 8 (Python Style Guide halten. (http://www.python.org/dev/peps/pep-0008/)

Plone 4.21				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
				„We have a few angels that follow the changes and each commit to see what happens to their favourite CMS! If you commit something REALLY sketchy, they will politely contact you, most likely after immediately reverting changes.“
	Strukturelle Mittel zur Durchsetzung der Qualitätskriterien	(+)	https://buildoutcoredev.readthedocs.org/en/latest/intro.html#id2 https://buildoutcoredev.readthedocs.org/en/latest/plips.html	- jenkins - Auto Tester - PLIPS (Review) - angels (s.o.)
	Transparenz des Entwicklungsprozesses	(0)	https://dev.plone.org/report/1	Ticketsystem zur Prüfung des Bearbeitungsfortschritt.
Dokumentation der Sicherheitsanforderungen und -ziele, Einsatzszenarien, Annahmen, Einschränkungen		(+)	http://plone.org/products/plone/security/overview	
Beschreibung aller Sicherheitsmechanismen auf Architektur-, Design- und Implementierungsebene		(+)	http://plone.org/products/plone/security/overview http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/sicherheit-und-arbeitsablaufe	
Prüfung / Test (Verifikation)				
Design-Review nachweislich		(+)	https://buildoutcoredev.readthedocs.org/en/latest/intro.html	„We have a few angels that follow the changes and each commit to see what happens to their favourite CMS!“
Code-Review nachweislich		(+)	https://buildoutcoredev.readthedocs.org/	Über einen pull request kann ein review

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				en/latest/intro.html	eingeholt werden.
Sicherheitstests nachweislich			(+)	http://pypi.python.org/pypi/plone.app.testing/4.2 http://developer.plone.org/testing_and_debugging/index.html	
Softwareentwicklung (Construction)					
Sichere Architektur	Modularisierung / Trennung der logischen Funktionen	Modularisierung / Trennung der logischen Funktionen	(+)	vgl. Kapitel 2.2.2.1 und 2.2.2.2	
		Trennung in verschiedene Sicherheitsniveaus möglich (Sandbox)	0	http://developer.plone.org/security/sandboxing.html	Sandboxing mit Python war möglich, ist jedoch gegenwärtig nicht gepflegt.
	Basistechnologie	Verwendung sicherer Bibliotheken	(+)	Testversion, http://plone.org/documentation/manual/installing-plone/installing-on-linux-unix-bsd/what-is-the-unified-installer	Es wird Python und das Zope 2 Framework verwendet. Die passenden Versionen werden mit dem „Unified Installer“ mitgeliefert. Es werden keine weiteren Bibliotheken verwendet.
		Ausrichtung an relevanten Standards	(+)	http://plone.org/products/plone/features/old/3.0/existing-features/standards-compliant-xhtml-and-css http://www.plone-nutzerhandbuch.de/webdav/index.html	- HTML5 - Rss - XHTML - CSS - WebDAV - WS-Security

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				http://plone.org/events/regional/past-events/nola05/collateral/web_services_presentation.pdf	- WS-Trust - SAML - SOAP - ... + test-driven development
	Integrationsfähigkeit	Web Services	(+)	http://packages.python.org/wsapi4plone.core/ http://plone.org/events/regional/past-events/nola05/collateral/web_services_presentation.pdf	- SOAP - WS-Security
		Payment	(0)	http://pypi.python.org/pypi/Products.PloneGetPaid/0.10.4 http://plone.org/products/collective.pfg.payment	Payment Module verfügbar
		Authentifizierung / Autorisierung	(+)	http://plone.org/products/webserverauth http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/authentifizierung/ldap http://plone.org/products/by-category/auth (Weitere durch Projekte)	Als Plugins verfügbar (OpenID, OAuth, LDAP) sowie Kerberos (WebServerAuth).

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				http://www.plone.de/ueber-plone/plone-broschure-als-pdf-datei	
	Skalierbarkeit des Systems	Nutzung von Caching Mechanismen	(0)	http://www.imn.htwk-leipzig.de/~mfrank/lehre/osem_10/MSchmeisser_MVogel_doku_caching.pdf http://www.plone.de/ueber-plone/plone-broschure-als-pdf-datei	
		Verteilbarkeit kritischer Komponenten zur Erhöhung Performance/ Verfügbarkeit	(+)	vgl. Kapitel 2.2.2.1	
	Auswahl und Reife der Erweiterungsmechanismen		(0)	http://developer.plone.org/getstarted/index.html#id5 https://pypi.python.org/pypi?%3Aaction=search&term=plone&submit=search http://developer.plone.org/getstarted/paste.html http://ploneapi.readthedocs.org/en/1.0.0-rc.1/	Erweiterungen über Python Packages.
Umsetzung von Sicherheitsan-	Rollen und Rechtekonzept	Vererbung von Rechten	(-)		Es konnten keine Informationen dazu gefunden werden.

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
forderungen		Definition von Gruppen	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/sicherheit-und-arbeitsablaufe/gruppen	
		Rollenbasierter Zugriff	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/sicherheit-und-arbeitsablaufe/rollen vgl. Kapitel 2.2.2.4	
		Attribut-basierter Zugriff	(-)		Es konnten keine Informationen dazu gefunden werden.
		Granularität der Zugriffssteuerung	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/sicherheit-und-arbeitsablaufe	
	Anpassbarkeit	Unterstützung mobiler Endgeräte	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/erscheinungsbild/theme-fuer-mobile-anwendungen	
		Nicenames / URL-rewriting	(+)	http://wiki.zope.org/zope2/ZopeAndApache	
		Personalisierung	(+)	http://www.plone-nutzerhandbuch.de/benutzer-rollen-und-gruppen/personalisierung/index.html http://www.plone.de/ueber-plone/plone-broschure-als-pdf-datei	
		Kontext-Sensitivität	(+)	demo.plone.de	

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
	Sprachvarianten	Inter-nationalisierung nach Best Practices realisiert	(+)	http://developer.plone.org/troubleshooting/unicode.html http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/internationalisierung	
	Sichere Daten-ablage	Ablage von Zu-gangsdaten	(+)	http://plone.org/products/passwordfield http://developer.plone.org/members/member_basics.html#exporting-and-importing-member-passwords http://docs.zope.org/zope2/zope2book/Security.html	- SSHA = Salt + SHA - Möglich über LDAP, Kerberos, Zope-Datenbank - Zope verwaltet User und bietet Ver-schlüsselung sowie Hashing der Pass-wörter out of the box - Plugins erhalten maximal Hashwerte der Passwörter von normalen User-Objekten
		Rücksicherungen konzeptionell vor-bereitet?	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/authentifizierung/emergency-user.html http://plone.org/documentation/kb/backup-and-recover-data-fs-in-linux/backup-plone/ http://www.imn.htwk-leipzig.de/~mfrank/lehre/osem_10/CBecker_Backup_Datensicherung_Vortrag.pdf	Backup über Script im laufenden Betrieb möglich. Rückspielen aller Daten zu Zeit-punkt x möglich.
		Vertraulichkeit und Integrität von Be-	(-)		Keine Hinweise zu diesem Feature ge-funden.

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		nutzerdaten bzw. Vorgangsdaten schützbar?			
		Systemdaten von fachlichen Daten separiert	(-)		Keine Hinweise zu diesem Feature gefunden.
	Suchmaschine	Reife der Suchmaschine	(+)	http://www.plone.de/ueber-plone/plone-4/erweiterte-suche-ubersichtlicher http://plone.org/products/collective.solr	
	Verwaltung der Inhalte	Redaktioneller Workflow definierbar	(+)	http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch/sicherheit-und-arbeitsablaufe/arbeitsablaeufe	"Die Redaktionsworkflows werden über Richtlinien bestimmt und können sehr weit konfiguriert werden" (vgl. Kapitel 2.2.2.4)
		Versionierung von Inhalten	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/umgang/versionierung.html	
		Wiederverwendung von Content zur Reduktion der Komplexität	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/artikeltypen/bild.html	
		Preview verfügbar	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/artikeltypen/gemeinsamkeiten.html	
	Authentifizierung	Registrierung	(+)	http://plonedemo.de	Konnte nicht tief geprüft werden. Kurzprüfung auf "http://plonedemo.de/"

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
					ergab keine sichtbaren Schwächen: - Nutzer ID nicht sichtbar - Bestätigungsmail mit URL zum Abschluss der Registrierung - URL hat Ablaufdatum
		Anmeldung	(+)	http://plonedemo.de	Konnte nicht tief geprüft werden. Kurzprüfung auf "http://plonedemo.de/" ergab keine sichtbaren Schwächen: - URL enthält keine NutzerID
		Brute-Force-Schutz (Passwort)	(+)	http://plone.org/documentation/kb/securing-plone http://developer.plone.org/sessions/login.html#locking-user-account-after-too-many-retries	Begrenzung der Login-Versuche.
		Passwort vergessen	(+)	http://plonedemo.de	Konnte nicht tief genug geprüft werden. Kurzprüfung auf "http://plonedemo.de/" ergab keine Schwächen: - URL enthält keine NutzerID
		Captcha	(+)	http://plone.org/products/plone-captchas	Via Erweiterung
	Sitzungsverwaltung	CSRF-Schutz	(0)	https://github.com/plone/plone.protect/	
		Secure- und HttpOnly-Flag bei Cookies	(0)	http://plone.org/documentation/kb/securing-plone	

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
Service Transition					
Installation / Deployment					
Vorbedingungen / Anforderungen	Root Access auf System		(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/installation/linux.html	Nicht zwingend aber empfohlen. Bei Installation ohne Root-Rechte funktioniert Plone dann auch mit Benutzerrechten. Dies ist aber in der Regel für Produktivsysteme bei Plone nicht wünschenswert.
	Transparenz der Abhängigkeiten; gibt es ein Build-System wie maven o.ä.		(+)	http://www.hasecke.com/plone-benutzerhandbuch/3.3.5/glossar.html	Buildout ist ein skript- und recipe-gesteuertes Buildsystem.
	Bibliotheken integriert, die möglicherweise Konflikte auslösen		(+)	Testinstallation	Es werden keine zusätzlichen Bibliotheken mitgeliefert oder vorhandene verändert (Der „Unified Installer“ installiert Python und Zope 2 sowie Plone).
Komplexität	Anzahl manueller Vor-/Nacharbeiten bei Konfiguration und Installation		(+)	http://plone.org/documentation/manual/installing-plone/installing-on-windows http://plone.org/documentation/manual/installing-plone/installing-on-linux-unix-bsd	Für die Installation sind keine zusätzlichen sicherheitsrelevanten Schritte notwendig.
	Verwendung von Standards	Automatisiertes Update durch Paketverwaltung	(0)	http://plone.org/documentation/manual/upgrade-guide http://www.hasecke.com/plone-benutzerhandbuch/4.0/installation/aktualisier	Es stehen Upgradescripte für die Aktualisierung bereit. Teilweise automatisiert über Buildout

Plone 4.21				
Kriterium		***Wert***	***Quelle***	***Anmerkung***
			ung.html	
	Sichere Default-Installation	(-)	http://plone.org/documentation/kb/securing-plone	Es wird ein zufälliges Administratorpasswort generiert. Secure Cookies und https Backend müssen manuell aktiviert werden
	Hinweis auf sicherheitsrelevante Einstellungen an Ort und Stelle	(+)	*.cfg, Testinstallation	Hinweise im üblichen Maß in den buildout.cfg Dateien
	Übernahme sicherheitsrelevanter Einstellungen bei Updates	(-)	http://www.hasecke.com/plone-benutzerhandbuch/4.0/installation/aktualisierung.html	Es waren keine Informationen hierzu auffindbar.
Anwenderdokumentation				
Lieferung von Guidelines, Tutorials		(+)	Plone-Benutzerhandbuch z.B. http://www.hasecke.com/plone-benutzerhandbuch/4.0/index.html und http://plone.org/documentation/manual/plone-4-user-manual sowie http://plone.org/products/plone/security/overview	Manuals zu: Installation, Design, Entwicklung sowie Sicherheit.
Qualität	Zielgruppenorientierung	(+)	http://www.plone.de/dokumentation/handbuecher	Z.B. End Users, Administrators, Developers

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
				http://plone.org/documentation/manual	
	Navigation und Suche		(+)	z.B.: http://www.hasecke.com/plone-benutzerhandbuch/4.0/index.html http://plone.org/documentation/manual/plone-4-user-manual	Mehrere Dokumentationen vorhanden. Anwendungsgebiete sinnvoll unterteilt. Jede Seite (=jeder Arbeitsschritt) einzeln adressierbar
	Mehrsprachigkeit		(-)	http://www.plone.de/dokumentation/	Manches in deutsch, aber nichts Sicherheitsrelevantes.
	Aktualität		(+)	http://www.plone.de/dokumentation/handbuecher	Meist Stand Oktober 2012 (Plone 4.2, aktuelles Feature Release)
	Vollständigkeit	Beschreibung aller Sicherheitsmechanismen mit Annahmen und Einschränkungen	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/ http://plone.org/documentation/kb/securing-plone http://plone.org/documentation/topic/Installation	Hardening Plone in der Knowledge Base: Use Case, nicht alle Mechanismen.
		Beschreibung aller sicherheitsrelevanten Einstellungen	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/ http://www.plone-entwicklerhandbuch.de/plone-entwicklerhandbuch	

Plone 4.21					
Kriterium			*Wert*	*Quelle*	*Anmerkung*
				http://developer.plone.org/security/index.html http://plone.org/documentation/kb/securing-plone	
		Beschreibung verschiedener Szenarien (z.B. externer ID-Provider)	(+)	http://hasecke.com/plone-benutzerhandbuch/4.0/ http://plone.org/documentation/kb/securing-plone http://plone.org/documentation/topic/Installation	Beschriebene Szenarien: SSO mit AD, robuste Plone/Zope-Installation, Installation unter Ubuntu, Debian, Mac OS X, Windows etc.
Service Operation					
Marktdurchdringung					
Internetpräsenz	Moderierte Foren mit Fokus Security (Anzahl)		(-)	http://plone.org/support/forums http://stackoverflow.com/questions/tagged/plone	Bei plone.com kein Forum speziell für IT-Sicherheit gefunden. Bei stackoverflow keine Message tagged mit plone und security.
	Akteure mit Fokus Security (Anzahl)		(+)	http://plone.org/foundation/about/team/SecurityTeam	Jeder Entwickler wird über Fehler in seinem Bereich informiert und hat diese zu beheben. Das Security Team überwacht diese.
Anzahl Installationen in Deutschland			(+)	http://plone.org/support/sites	Es werden 170 Sites für Deutschland gelistet

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
Behandlung von Änderungen					
Security Patches	Zeit bis zum Erscheinen		(0)	Für einen Zeitrahmen zum Bereitstellen von Patches oder Bugfixes gibt es nur einen Anhaltspunkt. Der vorhin bereits erwähnte, am 6. November 2012 bereitgestellte Patch wurde am 31.10.2012 offiziell gemeldet.	Keine Information. Kein öffentliches Bug Tracking. Nur eine Angabe aus AP1. You ask from when we knew it as a hole to the bug fix, but we don't keep records of that. My instinct is that that number would be on the order of 10 days, but we have nothing to back that up. (Matthew Wilkes, Plone security team)
	Transparenz der Begleitinformationen		(+)	http://plone.org/products/plone/security/advisories	Problembeschreibung vorhanden
Betrieb					
Logging	Revisionsfähigkeit	Integritätsgeschützt	(nf)		Keine Angaben gefunden
		Vollständig	(+)	http://www.hasecke.com/plone-benutzerhandbuch/4.0/umgang/workflow.html#fig-workflow-historie http://plone.org/documentation/faq/plone-logs	Logs sind vorhanden bei Verwendung des Publication Control System
	Loglevel konfigurierbar		(+)	http://developer.plone.org/testing_and_debugging/logging.html	Addons können verschiedene Loglevel nutzen
	Vertraulichkeit	Fachliche Logs von technischen Logs trennbar	(-)	http://developer.plone.org/testing_and_debugging/logging.html	Keine Angabe in der Dokumentation gefunden
		Verschlüsselung der	(np)		Konnte nicht geprüft/nachgewiesen

Plone 4.21					
Kriterium			***Wert***	***Quelle***	***Anmerkung***
		fachlichen Logs mit Bordmitteln möglich			werden.
		Rücksicherung offenbart keinen Klartext	(-)	http://plone.org/documentation/kb/backup-plone	Keine Angabe in der Dokumentation gefunden.
Einbindung in Systemmanagement	Laufzeitinformationen extrahierbar		(np)		Konnte nicht geprüft/nachgewiesen werden.
	Steuerung per Skript		(+)	http://plone.org/documentation/kb/startup-script-for-debian/tutorial-all-pages http://developer.plone.org/misc/commandline.html	Es können Skripte für verschiedene Funktionen genutzt werden.
	Steuerung per API		(-)	http://developer.plone.org/reference_manuals/external/plone.api/api/index.html	Keine Angaben in der Dokumentation.

Linkliste der wichtigsten Quellen

CMS	*Quelle*
TYPO3	http://typo3.org
	http://typo3.org/news/article/typo3-60-back-to-the-future
	http://wiki.typo3.org
	http://typo3.org/documentation/document-library
	http://typo3.org/teams/security
	http://buzz.typo3.org/teams/security
	http://wiki.typo3.org
	http://docs.typo3.org
	http://docs.typo3.org/typo3cms/SecurityGuide
WordPress	http://wordpress.org/
	http://codex.wordpress.org
	http://core.trac.wordpress.org
	http://learn.wordpress.com
Joomla!	http://www.joomla.org
	http://docs.joomla.org
	http://docs.joomla.org/Security
Drupal	http://drupal.org
	http://drupal.org/documentation
	http://drupal.org/security/secure-configuration
Plone	http://www.plone.de
	http://plone.org
	http://plone.org/products/plone/security/overview
	http://plone.org/products/plone/security/advisories/
	http://plone.org/documentation/
	http://plone.org/documentation/manual/installing-plone/
	http://www.plone.de/dokumentation/handbuecher
	http://www.hasecke.com/plone-benutzerhandbuch/3.3.5/

Stichwortverzeichnis

Begriff (Abkürzung)	*Bedeutung*	*Synonym*
Auslieferungssystem	Teil des CMS, das die Seiten der Website aus verschiedenen Informationsquellen zusammen so aufbereitet, wie der Benutzer sie sieht.	
Attribute Based Access Control (ABAC)	Attributbasierte Zugriffssteuerung: Vergabe von Zugriffsrechten abhängig von Attributwerten des Nutzers oder der Ressource.	
Benutzer	Menschen, die sich als Bürger oder Kunde die CMS-basierte Website ansehen, in ihr Informationen austauschen oder über sie Transaktionen auslösen.	Nutzer
Business-Process-Management System	Soft- und Hardware zur Verwaltung, Steuerung und Unterstützung von Diensten, die von den Mitarbeitern zur Lösung ihrer Fachaufgaben genutzt werden.	
Caching	Zwischenspeichern von Seitenteilen, die bereits einmal von einem Benutzer angefordert wurden. Da das Zusammensetzen (Rendering) von Seiten erhebliche Zeit benötigt, geht die Auslieferung dieser Seitenteile zukünftig schneller.	zwischenspeichern
Collaboration System	Soft- und Hardware zur Förderung der Zusammenarbeit verschiedener Benutzer. Klassische Möglichkeiten bestehen in der gemeinsamen Terminfindung, im Informationsaustausch.	
Content	Aufbereitete Informationen in verschiedenen Formaten (Text, Bilder, Filme, etc.).	Inhalt
Content Management System (CMS)	Soft- und Hardware zur Erstellung und Pflege von Inhalten auf Websites.	Web Content Management System
Dienstanbieter	Privatpersonen, Behörden, Firmen oder Organisationen im allgemein, die Informations- oder Transaktionsdienste über die mit dem CMS verwaltete Website anbieten.	Online-Dienstanbieter
Enterprise Content Management System (ECMS)	Soft- und Hardware zur Verwaltung, Strukturierung, Verteilung, Ablage von Inhalten. Ein ECM System unterstützt die Gesamtheit der Geschäftsprozesse des Unternehmens oder der Organisation allgemein.	
Exploit	Fertiger Code, der eine Schwachstelle eines Systems ausnutzt, um unberechtigt auf die zugrunde liegende Infrastruktur zuzugreifen.	
Metadaten	Zusätzliche Daten, die den Inhalten zugeordnet sind und ihn näher beschreiben.	
Modul	Software-Quellcode, der als zusammenhängende logische Einheit betrachtet wird.	Erweiterung, Extension, Add-On, PlugIn, Component
Nicename	Bezeichnungen (URLs) von Webseiten, die für Menschen verständlich sind.	
Portal	Website, die den Benutzern ermöglicht, Inhalte	

	personalisiert und situationsgerecht darzustellen.	
Protokoll	Die Kommunikation zwischen zwei IT Komponenten verläuft i.d.R. nach einem vordefinierten „Protokoll“, welches die Beschreibung der ausgetauschten Informationen und deren Abfolge ist.	
Records Management System	Soft- und Hardware zur Verwaltung von Akten, welche insbesondere den Aspekt der Revisionssicherheit technisch absichern.	
Redakteur	Mensch, der professionell mit Hilfe des CMS Inhalte der Website pflegt.	Online-Redakteur
Repository	Website, die Erweiterungsmodule zum Download vorhält und Mehrwertdienste dazu anbietet.	Verzeichnis
Retention period	Zeit, die ein Inhalt auf Grund fachlicher Bestimmungen aufbewahrt werden muss (darf).	Aufbewahrungsfrist
Veröffentlichen	Vorgang, um CMS-basierte Inhalte für den Benutzer zugänglich zu machen.	Publishing
Webseite	Gesamtheit aller Inhalte, die unabhängig von ihrem Format (z.B. Text, Bild) auf einer Seite im Webbrowser des Benutzers erscheinen.	
Website	Schaufenster im Internet, über das ein Dienstanbieter dem Benutzer seine Informationen und Geschäftsprozesse anbietet.	Internet-Auftritt, Internetpräsenz
WYSIWYG	Akronym für das Prinzip „**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et“ („Was du siehst, ist das, was du bekommst.“)	
zero-day-exploit	Ein Exploit, für den es vom Hersteller des Systems noch keine Information, keine Abhilfe, keine Fehlerbereinigung gibt.	

Abkürzungsverzeichnis

Abkürzung	*Bedeutung*
API	Application Programming Interface
ABAC	Attribute Based Access Control
BIT	Bundesstelle für Informationstechnik
BITV	Barrierefreie Informationstechnik-Verordnung
BSI	Bundesamt für die Sicherheit in der Informationstechnik
CAE	Content Application Engine
CMF	Content Management Frameworks (CMF)
CSRF	Cross Site Request Forgery“ (CSRF)
CVE	Common Vulnerabilities and Exposures
init	]init[AG für digitale Kommunikation
IIOP	Internet Inter-ORB Protocol
NVD	National Vulnerability Database
ORB	Object Request Broker
PDO	PHP Data Objects
REST	Representational State Transfer
RSS	Really Simple Syndication
SAGA	Standards und Architekturen für eGovernment-Anwendungen
SIT	Fraunhofer-Institut für Sichere Informationstechnologie
SSH	Secure Shell
TCP	Transmission Control Protocol
WYSIWYG	„What You See Is What You Get“
ZEO	Zope Enterprise Objects
ZODB	Zope Object Database